

Object-Oriented Distributed Database Design and Spatial Data Modeling

NRL Grant #N00014-97-1-G909
1997-2002

Final Report

*PI: Dr. Maria Cobb
Department of Computer Science & Statistics
University of Southern Mississippi
Hattiesburg, MS 39406-5106*

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

20021022 051

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

Public Reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimates or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188,) Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE 30-09-2002		3. REPORT TYPE AND DATES COVERED Final, 30/09/1997-29/06/2002	
4. TITLE AND SUBTITLE Object-Oriented Spatial Database Design and Spatial Data Modeling				5. FUNDING NUMBERS N00014-97-1-G909	
6. AUTHOR(S) Maria Cobb					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Southern Mississippi Box 5157, Hattiesburg, MS 39406-5157				8. PERFORMING ORGANIZATION REPORT NUMBER 64-6000818	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Research Laboratory				10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES					
12 a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12 b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Work for this grant enabled the advancement of the state of research for areas of fuzzy spatial data modeling, uncertainty and conflation. Specific published work supported by this grant included a range of topics, including: fuzzy spatial relationship refinements, spatial query interfaces, uncertainty in distributed spatial information systems, spatial relationship querying, distributed spatial object-based systems, spatial indexing, distributed conflation model and issues, and an integrated image change detection/conflation algorithm. Special journal issues co-edited by the grant recipient on "Spatial Data Management," "Uncertainty in Geographic Information Systems and Spatial Data" and "Distributed Object-Oriented Systems" resulted in newly published works from some of the foremost researchers in the area, including Michael Goodchild, Peter Fisher, Hans Guesgen, Douglas Schmidt and others. All publications listed are included in this report package, and the reader is referred to the actual publications for details of the topics listed above.					
14. SUBJECT TERMS conflation, spatial querying, fuzzy spatial relationships				15. NUMBER OF PAGES 4	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UU		

Introduction

This grant was originally issued for the investigation of state-of-the-art technology component-ware for distributed databases utilizing CORBA, as well as the development of a fuzzy spatial data model to provide a way of handling imprecision and uncertainty in spatial data. Specifically, the grant work was to augment ongoing development of NRL's Geospatial Information Database (GIDB) in the realm of Object Request Brokers (ORBs) and the GemStone Object-Oriented Database Management System (OODBMS).

Soon after the work was initiated, however, NRL began to investigate alternate implementation architectures for the GIDB relative to its future stability and viability. Major re-design decisions were made, including: (1) a move from the Smalltalk programming language to Java, (2) replacement of GemStone OODBMS with Ozone OODBMS (Java-based), and (3) replacement of CORBA-based architecture with Java Remote Method Invocation (RMI)/applet technology. These decisions have since proven to be invaluable to the development and marketability of the GIDB; however, because this new direction made much of the proposed grant work in CORBA and related implementation issues unnecessary, and in consultation with the NRL principal scientist, the concentration of work was redirected to the areas of fuzzy spatial data modeling and conflation.

After the original three years were completed, a no-cost extension was requested. This extension funded a Ph.D. scientific computing student who graduated August 2002. A copy of the student's dissertation on conflation is included with this report, along with all other grant-related publications.

Summary of Findings

Work for this grant enabled the advancement of the state of research for areas of spatial data modeling, uncertainty and conflation. Special journal issues co-edited by the grant recipient on "Spatial Data Management," "Uncertainty in Geographic Information Systems and Spatial Data" and "Distributed Object-Oriented Systems" resulted in newly published works from some of the foremost researchers in the area, including Michael Goodchild, Peter Fisher, Hans Guesgen, Douglas Schmidt and others.

Specific published work supported by this grant included a range of topics. Some of these are: fuzzy spatial relationship refinements, spatial query interfaces, uncertainty in distributed spatial information systems, spatial relationship querying, distributed spatial object-based systems, spatial indexing, distributed conflation model and issues, and an integrated image change detection/conflation algorithm.

All publications listed are included in this report package, and the reader is referred to the actual publications for details of the topics listed above.

Results

This grant has resulted in basic research into fuzzy spatial data models and conflation of spatial data. It has resulted in the publication of eight conference papers, three book chapters, four journal articles and three edited journal issues. It has also funded one Scientific Computing Ph.D. student (graduated August 2002). Grant funding also enabled the USM PI to perform the following service activities to promote research and knowledge in relevant subject areas:

- ❖ Co-organized and chaired panel session, "Can Statistical and Fuzzy Sets Approaches Complement Each Other in Dealing with the Problem of Uncertainty in Spatial Data?" at the International Symposium on Spatial Data Quality '99, Hong Kong (1999).
- ❖ Organizing co-chair and chair of four special conference sessions:
"Uncertainty Management in Spatial Data" sessions held at:

Information Processing and the Management of Uncertainty (IPMU '98), Paris, France (1998), and

International Conference on Fuzzy Systems (FUZZ-IEEE '98), Anchorage, AK (1998).
- ❖ "Distributed Object-Oriented Systems"--two sessions
1998 International Conference on Parallel and Distributed Computing and Systems (PDCS '98), Las Vegas, Nevada (1998).
- ❖ "Spatial Data Computation"--three sessions
First Southern Symposium on Computing (SSC '98), Hattiesburg, MS (1998).

List of Publications

1. Cobb, M., K. Shaw, "Distributed Object-Oriented Systems," *Parallel and Distributed Computing Practices*, (2001).
2. Cobb, Robinson, Petry and Shaw, "Uncertainty in Geographical Information Systems and Spatial Data," *International Journal of Fuzzy Sets and Systems* (2002).
3. Cobb, M., K. Shaw and F. Petry, "Spatial Databases," *Informatica*, Vol. 23, No. 2 (1999).
4. Chung, M., R. Wilson, K. Shaw, F. Petry and M. Cobb, "Querying Multiple Data Sources via an Object-Oriented Spatial Query Interface and Framework," *Journal of Visual Languages and Computing*, (2001) 12, pp. 37-60.
5. Cobb, M., F. Petry and K. Shaw, "Fuzzy Spatial Relationship Refinements Based on Minimum Bounding Rectangle Variations," *Intl. Journal of Fuzzy Sets and Systems*, Vol. 113, No. 1, pp. 111-120, July 2000.
6. Cobb, M., Miyi J. Chung, Vincent Miller, Harold Foley III, Frederick E. Petry, Kevin B. Shaw, "A Rule-Based Approach For The Conflation of Attributed Vector Data," *GeoInformatica*, 2(1), 7-35 (1998).
7. Cobb, M., H. Foley III, R. Wilson, M. Chung and K. Shaw, "An OO Database Migrates to the Web," *IEEE Software*, 15(3), 22-30 (1998).

8. Cobb M., H. Foley, F. Petry, K. Shaw, "Uncertainty in Distributed and Interoperable Spatial Information Systems" invited book chapter for *Recent Issues on Fuzzy Databases*, eds. Gloria Bordogna and Gabriella Pasi, pp. 85-108, Physica-Verlag, 2000.
9. Chung, M., R. Wilson, R. Ladner, T. Lovitt, M. Cobb and M. Abdelguerfi, "The Geospatial Information Distribution System (GIDS)," in: *Succeeding with Object Databases*, eds. Akmal B. Chaudhri and Roberto Zicari, pp. 357-378, 2000.
10. Shaw, K., M. Chung and M. Cobb, "Migration Process and Considerations for the Object-Oriented Vector Product Format to ObjectStore Database Management System," in: *Object Databases in Practice*, M.E.S. Loomis and A.B. Chaudhri, Prentice-Hall, 234-253 (1998).
11. Yang, H., M. Cobb, D. Ali, S. Rahimi, F. Petry, K. Shaw, "Fuzzy Spatial Querying with Inexact Inference," NAFIPS 2002, New Orleans, LA (2002).
12. Yang, H., M. Cobb and K. Shaw, "A CLIPS-Based Implementation for Querying Binary Spatial Relationships," NAFIPS 2001 Proc., July 25-28, 2001, Vancouver, Canada, pp.2388-2393.
13. Cobb, M., M. Chung, R. Wilson, K. Shaw and F. Petry, "Spatial Data Mining Using Fuzzy Logic in an Object-Oriented Geographical Information Database," *Proc. World Multiconference on Systemics, Cybernetics and Informatics*, SCI/ISAS '99, Orlando, FL, 117-124 (July 31-August 4, 1999).
14. Chung, M., R. Wilson, K. Shaw and M. Cobb, "Distributing Mapping Objects with the Geospatial Information Database," (invited) *Proc. Intl. Symposium on Distributed Objects and Applications*, Edinburgh, Scotland, 77-85 (Sept. 5-6, 1999).
15. Shaw, K., M.J. Chung, R. Wilson, R. Ladner, M.A. Cobb, T. Lovitt, "An Object-Oriented Database Approach for Urban Warfare," *Proceedings of Urban and Regional Information Systems Association*, URISA '99, Chicago, IL, 272-283 (August 21-25, 1999).
16. Cobb, M., F. Petry and K. Shaw, "Uncertainty Issues of Conflation in a Distributed Environment," *Proceedings of GIS/LIS '98*, Fort Worth, TX, 436-448 (Nov. 10-12, 1998).
17. Wilson, R., M. Chung, T. Lovitt, B. Ray, M. Cobb and K. Shaw, "Design of a Java Interface to a Smalltalk OO Mapping Database Utilizing CORBA," *Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS '98)*, Las Vegas, NV, 60-63 (Oct. 28-31, 1998).
18. Wilson, R., M. Cobb, M. Chung and K. Shaw, "A Spatial Indexing Framework Using a Quadtree Organization for Geographic Data Storage and Retrieval," *Proceedings of the First Southern Symposium on Computing (SSC '98)*, Hattiesburg, MS (Dec. 4-5, 1998).

The University of Southern Mississippi

A DISTRIBUTED MOBILE AGENT CONFLATION MODEL
UTILIZING AN IMAGE CHANGE DETECTION ALGORITHM

by

Huiqing Helen Yang

Abstract of a Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

August 2002

ABSTRACT

A DISTRIBUTED MOBILE AGENT CONFLATION MODEL UTILIZING AN IMAGE CHANGE DETECTION ALGORITHM

by Huiqing Helen Yang

August 2002

The Geographic Information System (GIS) is an integrated technology that incorporates concepts from computer graphics, spatial modeling, and database management. The distributed intelligent mobile agent technique, which successfully incorporated more powerful technology, is becoming an important issue in Geographical Information Systems. Within the distributed environment, the compatibilities and consistencies are mainly concerned issues. The "conflation" is an important and challenging technique to handle these issues.

Generally, conflation means to combine information from different sources and then to produce better information. Up to now, the conflation consideration has become much broader in GIS research fields. Many efforts have been made. However, conflation is still a challenging research field due to the complexity of real applications. Seen from the existing conflation paradigms, the conflation algorithms have been ad hoc, designed for specific purposes. The focus of the dissertation is placed mainly on processing the vector-based conflation problems.

Considered the vulnerability to deal with time in existing conflation algorithms, the endeavor of the dissertation is to explore the ways in which conflation capabilities can be augmented with the aid of change detection techniques. A general and flexible conflation model is proposed for the distributed mobile agent systems.

Based on the model, over time considerable effort is specially spent on the development of image change detection algorithm. Since image change detection, like many other applications in GIS, requires to handle fuzziness and uncertainty, an innovation is investigated – it is an intelligent approach in which the issue associated with fuzziness and uncertainty has been tackled by introducing a Certainty Factor. A hierarchical structure for the fuzzy inference is figured out. Theoretical analysis and real image evaluation show that it can provide significant results.

COPYRIGHT BY
HUIQING HELEN YANG
2002

The University of Southern Mississippi

A DISTRIBUTED MOBILE AGENT CONFLATION MODEL
UTILIZING AN IMAGE CHANGE DETECTION ALGORITHM

by

Huiqing Helen Yang

A Dissertation

Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved: Dr. Maria Cobb

Director

Dean of the Graduate School

August 2002

ACKNOWLEDGMENTS

This dissertation details a tremendous amount of work, which could have never been completed by an individual. Many others have contributed their time, knowledge, talent and support which allowed for development and assembly of the finished product. This is to thank all of those who have assisted me on the path towards this dissertation, both during and before my study at the University of Southern Mississippi (USM).

In 1988, I came to America after earlier careers which included a master degree, and an ABD, and several years teaching experience as a full-time instructor. It is the America Dream that made it impossible to complete my Ph.D dissertation. However, the methodological training I had has been of great benefit to me in academic field. At this particular time, I would like to express my heartfelt gratitude to Professor Zhonghuai Zhang, a Professor at Taiyuan University of Technology in China, who had be advising me at teaching and research. The high standards that he exemplifies in his roles as an educator, mentor, and researcher have established a firm foundation upon which I will base my research and teaching. My current research agenda continues to be positively influenced by the fundamental training.

During studying for my second master degree in Computer Science at the University of Southern Mississippi, and pursuing for the Ph.D, I have had the great fortune of benefiting from the wisdom, encouragement, and friendship of many mentors.

First and foremost, I wish to acknowledge my dissertation work to Dr. Maria Cobb who initially approached me on this topic, and provided much of the framework for my thinking. I am extremely grateful for the opportunity to have had her as my advisor for the past few years, with special thanks to her delicate guidance, gracious mentoring and financial support. I learned so much more about various approaches to research than I would have before. Her insight and influence have always kept me pointed in the right direction. I am forever indebted to my advisor, Dr. Maria Cobb who is the source of all wisdom in this worldly life.

Next, I would like to extend my special thank to Dr. Dia Ali, one of my committee members and also my advisor for my second master degree, who went above and beyond the "call of duty" by generously affording me his time, patience, kindness, thinking, and input. It is him who initially led me to pursue studying for Ph. D at USM. Without his inspiration, this study would not be possible. I am really grateful for his valuable advice and kind help beyond the academic study. Moreover, I cannot forget his assistance and delicate guidance in my job hunting. He is the gentlest person that I have ever met, and one of the kindest.

I am also sincerely grateful to the rest of the committee members, Dr. Jiu Ding, Dr. Joseph Kolibal, Dr. Louise Perkins and Dr. Fred Petry for their time and energy in helping me improve this dissertation. I have had the pleasure of learning from them in other ways as well.

Particularly, I wish to truly express my appreciation for Dr. Joseph Kolibal who devoted much time and patient expertise to the development of the Linux-based L^AT_EX package for the dissertation presented within this volume. His willingness to share of his time in helping prepare my format give me the opportunity to gain new skills. Also I wish to express much appreciation for Dr. Jiu Ding who contributed in the classroom with excellent teaching and assisted me in mathematical knowledge. Sincerely, I would like to thank Dr. Perkins for her comments and suggestions. Her encouragement and willingness to participate in my course learning process are also appreciated. The time, patience, and teachings that they have shown and given me will enable me to pursue a higher level in the field of education.

At the same time, I am grateful to Dr. Fred Petry from Tulane University for his support. The support of the Naval Research Laboratory Base Program is also appreciated.

Finally, a hearty thank goes to my family: my sweet daughter Angela, and my husband John, who have endured with me much of the trials and long hours necessary to make it all possible - thank you for being my support. Also, I acknowledge, appreciate, and return the love and support of my parents who has been unconditionally supportive of my academic and non-academic exploits over the years.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
LIST OF ILLUSTRATIONS	vi
LIST OF TABLES	vii
LIST OF ABBREVIATIONS	viii
 1 INTRODUCTION	 1
2 BACKGROUND	4
2.1 Data Conflation Issue and Related Work	4
2.2 Conflation Classification	7
2.3 A General Conflation Method for Vector Data	9
2.4 Change Detection Issue and Related Work	10
2.5 Image Change Detection Methods	10
2.6 Change Detection Consideration	12
 3 A FLEXIBLE CONFLATION MODEL	 14
3.1 Data Models in GIS and Their Conversion	14
3.2 A Mobile Agent Prototype Model for Conflation	16
3.3 A Raster-Based Vector Conflation Model	18
3.4 Significance of the New Model	20
 4 A VECTOR-BASED CONFLATION SCHEME	 22
4.1 Conflation Components	22
4.2 Intelligent Conflation Algorithms	25
 5 IMAGE CHANGE DETECTION	 30
5.1 Basic Concepts for Raster Data	31
5.2 Basic Concepts of Fuzzy sets	34
5.3 A Hybrid Approach for Image Change Detection	37
 6 EVALUATION AND RESULT ANALYSIS	 51
6.1 Theoretical Analysis of Image Change Detection	51
6.2 Evaluation via Real Image Data	56

6.3	Real Implementation Consideration	59
6.4	Summary of the Hybrid Approach	61
7	CONCLUSION	63
7.1	Main Contributions	63
7.2	Open Research Issues	65
7.3	Future Work	65
	BIBLIOGRAPHY	67
	INDEX	71

LIST OF ILLUSTRATIONS

Figure

2.1	Unmatched Edge	4
2.2	Conflation Classification Based on Geometry	7
2.3	Conflation Classification based on Data Model	8
2.4	A General Conflation Process	9
3.1	Examples of Raster and Vector Data	15
3.2	A Distributed Mobile Agent Model	16
3.3	A Conceptual Model for Distributed Conflation	18
3.4	A Raster-based Vector Conflation Model	19
4.1	Feature Matching Types	27
5.1	Discrete Raster Image	31
5.2	Histogram of an Image	33
5.3	Gaussian Function	36
5.4	Membership Functions of an Image	45
5.5	A Hierarchical Model for Fuzzy Inference	49
6.1	Histogram Analysis	52
6.2	Membership Functions for Analysis	53
6.3	A Fuzzy Uncertain Inference for Image I_1 and I_2	54
6.4	A Fuzzy Uncertain Inference for Image I_1 and I_3	55
6.5	Pre-detection for Real Image Data	57
6.6	Image Feature for Post-detection	58
6.7	Membership Functions 1 of Intelligent Detection	59
6.8	Membership Functions 2 of Intelligent Detection	59
6.9	Fuzzy Inference of Real Image Data	60
6.10	Summary of the Hybrid Detection Method	62

LIST OF TABLES

Table

4.1	An Example of Attribute Table	29
5.1	Groups of the Fuzzy Moments for CF	48

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligent
AM	-	Agent Manager
CA	-	Conflation Agent
CF	-	Certainty Factor
DMAP	-	Digital Mapping, Charting and Geodesy Program
GIS	-	Geographical Information System
IFOV	-	Instantaneous Field of Views
MF	-	Membership Function
NRL	-	Naval Research Laboratory
PCA	-	Principal Components Analysis
QA	-	Querying Agent
QM	-	Queue Manager
RA	-	ROI Agent
ROI	-	Region Of Interest
USGS	-	United States Geological Survey
USM	-	The University of Southern Mississippi
VPF	-	Vector Product Format

Chapter 1

INTRODUCTION

The Geographic Information System (GIS) is a computer-based information system that can deal with collecting, modeling, managing, analyzing, and integrating spatial and non-spatial data in geographic applications. It has been experiencing a steady and unprecedented growth in terms of the general interests, theory development, and new applications in the last decade or so. Up to date, GIS has matured to serve important roles for many academic disciplines, government organizations, and commercial enterprises.

The handling of geographic information has always raised an issue of scientific nature. In general, GIS is an integrated technology that combines concepts from computer graphics, spatial modeling, database management, and remote sensing. However, no single technology can by itself fully meet the requirements of a GIS application [3]. There is a wide agreement in the GIS community that the future success of GIS technology will depend to a large extent on incorporating more powerful analytical capabilities.

The distributed intelligent mobile agent technique is becoming an important issue in GIS research areas due largely to the following technological advantages:

- During the past 20 years, the computer architecture has moved from stand-alone systems to local and wide-area networks. A natural extension to this development is to study the distributed technology on geographic information. Recently, the trend for most types of information systems has been a move to a more loosely coupled, distributed nature. The typical GIS applications are GeoChange [14] and Alexandria Digital Library Project [16], which are already underway to provide users access to geographic data cross wide area networks.
- Recent decades have been characterized by the generalized application of artificial intelligent technology to improve the system's capabilities. We have witnessed a rapid growth in the number and variety of applications of intelligent technology, ranging from real-time control systems, and decision support systems to business systems and the World Wide Web field. Since many applications in GIS involve human expertise and knowledge, which are invariably imprecise, incomplete, or

not totally reliable, the intelligent technology becomes a potential tool to solve GIS problems.

- Moreover, the emerging of a quiet GIS industry revolution shows that GIS is going mobile.

Recently, the distributed intelligent mobile agent systems have been developed for different purposes and applications. The project in [9] presents an autonomous agent system that utilizes the distributed intelligent agent techniques to handle the geo-spatial data from multiple heterogeneous sources.

Within the distributed environment, the data updating, integrating, and sharing will be concerned as central issues in GIS. With these issues, new problems such as data structure incompatibility, accuracy incompatibility, scales of measurement incompatibility, or inconsistencies, etc., are arising. The most common reasons for such problems are that different spatial datasets from different sources might use different project systems, different scale and accuracy, and so on. For dealing with such problems, the important and challenging technique is called conflation.

Generally, the conflation is regarded as the combination of information from two or more digital maps to produce a third map that is better than either of its component sources. The objectives of conflation include increasing spatial accuracy and consistency, and updating or adding new features into datasets, updating or adding more attributes that associate with the features in datasets, etc.

Seen from the regular conflation paradigms, one of the vulnerabilities is lack of capability to deal with time. If an existing resource of information fails to suit the requirements because it is out-of-date, a process of detecting and then updating may be more effective than a total reconstruction. Thus, it is important to identify differences in the images in terms of real changes.

Traditionally, changes to geographical phenomena have been derived from a temporal reference frame. Since time is difficult to be formalized [22], to date, no single model in a GIS, which includes time, has been adopted. In recent years, by integrating more powerful techniques, GISs offer many possibilities for improved treatment of time and changes. Recent research results have demonstrated that the change detection can be done from satellite images or other images, accompanied by visual interpretation of

differences. Therefore, as a potential technique, image change detection encourages us to make an attempt on improving conflation capabilities.

The objective of image change detection is to detect changes in images over time. Augmenting conflation by utilizing satellite images is the motivation of the dissertation. The rest of the dissertation is organized as follows. The following Chapter gives an overview of related work, and shows current methodologies to deal with conflation problems and image change detection. Considered a vector GIS, a more general and flexible conflation model is developed for a distributed system in Chapter 3. Based on the model, two important contributions are emphasized in the following chapters. Chapter 4 mainly introduces a vector-based conflation scheme. A hybrid approach for image change detection is proposed in Chapter 5. It is evaluated by using real raster images in Chapter 6. In the conclusion, the open research issues are discussed and future research ideas for possible improvements of the algorithm are provided.

Chapter 2

BACKGROUND

Geographic information provides the basis for many types of decisions. With respect to conflation, the merging of geographical data allows for manipulation, analysis, and comparison within and between multiple geographical databases.

Too often, the conflation methods have been ad hoc, designed for specific purposes. In this chapter, the overview of previous work related to data conflation is firstly given. Then, based on the geometric view and GIS data models, the conflation problems are classified into different groups so as to understand the conflation problems well. A general conflation method for vector data is summarized. For the purpose of augmenting conflation, image change detection issue is addressed. And then current and potential approaches in the image change detection are surveyed.

2.1 Data Conflation Issue and Related Work

When geo-spatial data from different sources are combined and shared, incompatibilities and/or inconsistencies in geographical data occur. Figure 2.1 shows a problem arising from merging data digitized from adjacent map sheets.

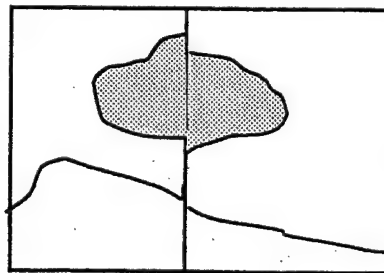


Figure 2.1: Unmatched edge occurring in a cartographic image.

However, the incompatibilities may be not only structural but also semantic in nature. For example, the attributes for representing the same values might be defined differently in different sources, which may include different names or different domains for their associated value, e.g. float versus integer.

Therefore, it is quite a challenge to preserve the semantics inherent in the component datasets. The important technical issue to handle this situation is "conflation". The term "conflation" is used to refer to the integration of data from different sources. Conflation is the complex process of recognizing and removing inconsistencies (or discrepancies) in geographical feature data that are stored in multiple databases.

The history of map conflation goes back to the early mid-1980s. The first clear development and application of an automated conflation process occurred during a joint United States Geological Survey (USGS) Bureau of the Census project designed to consolidate the agencies' respective digital map files of U.S. metropolitan areas [51]. The major concern of conflation was to eliminate the spatial inconsistency and improve the spatial accuracy of maps. The implementation of a computerized system for this task provided an essential foundation for much of the theory and many of the techniques used today.

Since that time, others, including numerous agencies, institutions and commercial GIS vendors, have developed and implemented conflation tools within their applications. The conflation consideration in GIS has become much broader. Some practical examples are provided below.

The EDNA project of the USGS investigated two different processing approaches for conflation in order to share common attributes from separate coverages [58]. Atlas has developed a conflation engine that automatically finds the common elements in two vector data sets; the engine is written in C++ and uses an object oriented topological GIS infrastructure [57]. Due to a significant difference between the TIGER and DALIS Projects geography, the project [11] provides a method that conflates the Census 1990 Block coverage to the DALIS base. It demonstrates that it is more important to be able to effectively relate all data sources with other valuable attribute information files such as census-attributed data.

Right now automatically conflating digital map data has presented a rich set of computational challenges. Based on geo-statistical techniques, some primary conflation paradigms are presented in the literature, and have been proven to be very effective for regular data such as street networks. In [42], a geo-statistical method for combining data of different sources is presented, and its reliability is analyzed. However, it is surprised that very little progress has been made since the first successful implementation of a system based on geo-statistical techniques. One of the main reasons is the complexity and uncertainty of GIS problems.

As GIS becomes more complex and intelligent, artificial-intelligent-based methods will become a cost-effective approach for improving its problem-solving abilities. More recently, researchers have turned to fuzzy logic and other intelligent methods for handling reasoning abilities under conditions of uncertainty to help solve general conflation problems. This approach certainly shows greater promise for producing a wider range of acceptable results; however, models for implementation have still been limited. The following gives a brief survey of some related work.

The Air Force Rome Laboratory Multiple Data Base Integration and Update system [35] reviews the various technical challenges encountered during automated vector conflation and all-source updating of a unified vector data baseline, and highlights its functions by taking the advanced technologies like Artificial Neural Networks. In [7] a rule-based approach for conflation of attributed vector data is introduced, which was performed within the context of the Digital Mapping, Charting & Geodesy Program (DMAP) at the Naval Research Laboratory (NRL), Stennis Space Center, Mississippi. A knowledge-based system to handle uncertainty issues of conflation in a distributed environment is developed in [8]. The paradigm is based on a hierarchical rule-based system that utilizes techniques for reasoning under uncertainty. These kinds of models proposed are to overcome some of the more difficult feature matching cases encountered in more generic, irregular geographic features.

Although a little improvement is made, the issues associated with conflation have serious consequences for many applications, which range from combining digitized topographic maps, to edge-matching of misfit data across boundaries, to combining information from different sensors in remote sensing. It is no doubt that users are faced with extensive duplication of efforts and unnecessary cost without the ability to conflate/integrate data from different sources.

As a result, conflation typically is needed because:

- Users wish to update their information without losing legacy data which may not be included in the new information;
- One source may be more accurate with respect to information such as position and attribute; and,
- One source contains information missing in another, such as additional features, feature attributes or even entire coverages.

2.2 Conflation Classification

Conflation means to combine information from different sources and then to produce better information. From the geographic point of view, GISs are spatial reference systems in which spatial data are generally treated in two dimensions, i.e. longitude and latitude directions. The spatial reference system provides the geometric basis to connect the different sources. The combinations of different sources in the spatial reference systems may result in three different versions of conflation. More details are given as follows:

- Data conflict when combining the same type layers of a region;
- Data overlay when combining the different type layers of the same region; and,
- Edge matching when combining the same type layers of neighboring regions.

A general conflation classification is shown in Figure 2.2.

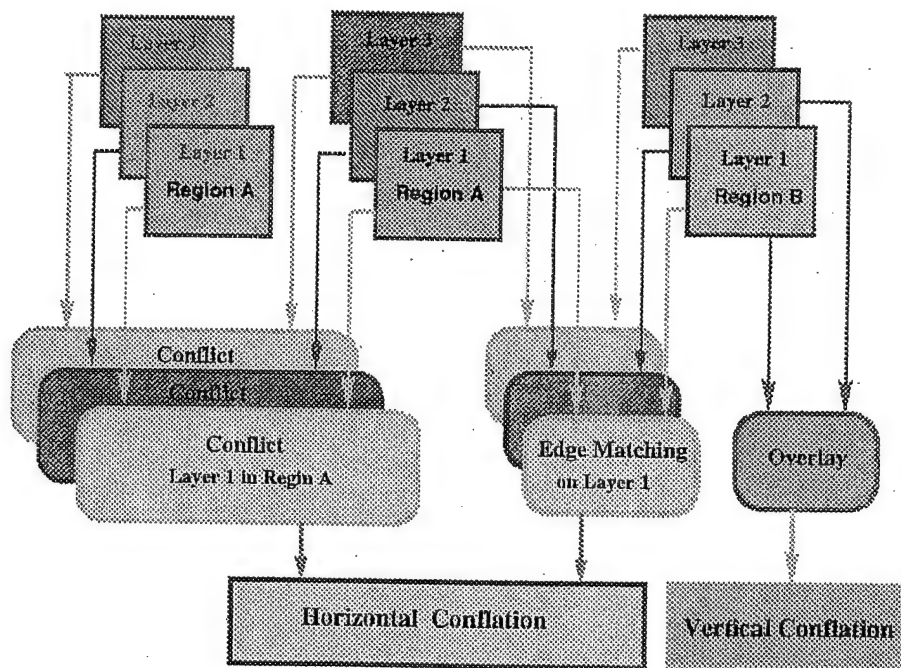


Figure 2.2: A general conflation classification based on spatial geometry

The objective of the horizontal conflation is to eliminate the spatial feature position and attribute discrepancies that exist in the common area of two sources [64]. The

vertical conflation is actually a process to perform an overlay operation on the datasets from different layers of the same region. It requires that the two sources are identical. Also, since none of sources is perfectly error-free, the apparent change includes some amount of errors mixed with the actual change.

The typical applications of the vector-based conflation include [64]:

- **Spatial Discrepancy Elimination:** It is a global adjustment process for spatial feature coordinates in order to eliminate the feature position discrepancies. This application exists in map edge-matching, map compilation, and etc.
- **Spatial Feature Transfer:** In this process, the common features from different sources are recognized, and flagged; new features can be added into the old source, or old coordinates can be updated. The main applications exist in the GIS spatial data updating.
- **Attribute Transfer:** Usually, it is used to transfer the lower spatial accuracy source to the higher accuracy source, or old version to new version.

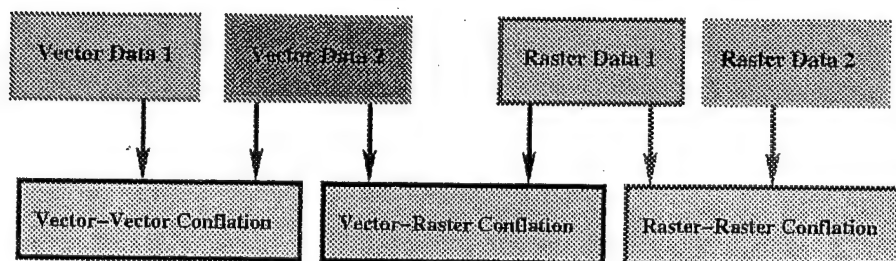


Figure 2.3: A conflation classification based on GIS data model

Based on the data models in GISs shown in Figure 2.3, the conflation can be classified as vector-vector, vector-image, and image-image conflation.

The most common conflation problem is the vector-vector conflation since many products in GIS are based on the vector data. Conflating vector data with imagery is also an important aspect in GIS applications. It involves many digital image-processing technologies such as image feature extraction, shaping, re-sampling, model recognition, and so on. Image to image conflation involves more image matching technologies. Usually, it is performed by specialists.

2.3 A General Conflation Method for Vector Data

The goal of conflation is to have the “best” information for a given area. Seen from the previous views, there are many algorithms that have been developed to solve different practical conflation problems. For vector database users, the objective of conflation is to combine different source coverage into one, and to reconcile the best features geometries and their related attributes from the different source coverages. The general conflation processing steps can be shown in Figure 2.4.

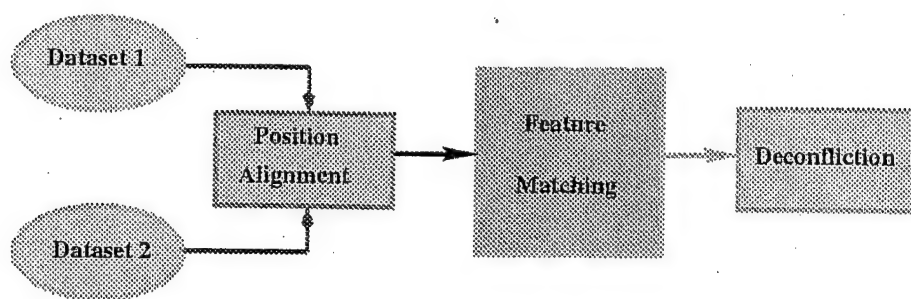


Figure 2.4: Steps involved in a general conflation process

- **Positional re-alignment**

Since several coordinate systems such as the Universal Transverse Mercator, Global Position System, and Local Cadastral System may be used in a distributed system, integrating information from different sources requires transformation of the coordinate system of the source to the one adopted for the specific spatial database in use. Positional re-alignment is a mathematical procedure in which previously identified matching features are brought into spatial agreement.

- **Feature matching**

Simply and perhaps somewhat obviously stated, feature matching involves the identification of features from different maps as being representations of the same geographic entity.

In fact, feature matching can be considered as a type of classification problem. That is, we are trying to determine whether one feature belongs to the same “class” as another. This type of problem can be handled through theories of evidential reasoning uncertainty such as fuzzy logic [29] or Dempster-Shater theory [54].

- **Deconfliction**

It is a process in which contradictions in a matching pair's attributes and/or values are resolved.

2.4 Change Detection Issue and Related Work

Change detection has been widely studied in the field of computer vision, remote sensing, and image processing. Changes over time in a geographical area are particularly important in such applications as deforestation, archeology, environmental monitoring, urban planning and development, damage assessment, and so on.

Image change detection is a process of identifying differences in the state of an object or phenomenon by comparing the images at different times. Since the early 1970s, satellite images in digital form have been available. This vast amount of satellite data offers unique possibilities of comparing images from an earlier date with new ones. In [44] several satellite image change detection techniques are investigated, and results show that the satellite image change analysis can provide good information. Moreover, remarkable advances in remote sensing technology have motivated many researchers to exploit change detection approaches and techniques for a variety of purposes, such as identifying buildings, monitoring regional changes, land use, wide area surveillance over a long period of time, and etc. [50]–[55] [21].

In the recent decade, change detection from remote sensing images has emerged as one of the most active and fruitful areas in GISs. Such applications can be found in many GIS, for example, a change detection approach for linear features in aerial photographs in [50], digital change detection techniques for civilian and military in [10] and for wide area surveillance in [37]. Specifically, the paper [47] investigated the applicability of image detection techniques for the purpose of emergency response by utilizing remotely sensed bi-temporal imagery data.

The successful application of the change detection techniques from images, and the emphasis on near real-time and new information encourage us to use change detection techniques for augmenting conflation.

2.5 Image Change Detection Methods

There are many methods and techniques for the detection of changes, which have been developed by mathematicians, geographers, philosophers, and computer scientists.

The basic and commonly used techniques are briefly described as follows [10, 26]:

Image Deviation: The image deviation is a basic and simple method for analyzing long time series data. With image deviation, the change areas are identified by contrast to a long-time average. The possibilities would be to produce a mean image over the whole series, to examine trends in environmental changes or detect significant anomalies from general trend.

Image Differencing: The image differencing is one of the simplest techniques for detecting changes between two images. With this, each pixel from an image is subtracted from its corresponding pixel in another image. The resultant image represents the change between the two different times.

Cross-classification: It is a procedure used to compare two images by calculating the logical AND of all possible combinations on the two classified input images. The aim is to evaluate whether areas fall into the same class in the two dates or whether a change to a new class has occurred.

The basic change techniques actually are two different kinds of change detection techniques: differencing and classification. They provide a basis for later change detection techniques.

Since the basic methods are used in small area information, for dealing with larger area information, some efficient algorithms are developed for specific purposes. In [48], three different methods for different motivations are introduced, that is, a linear change detection for small changes, non-linear change detection for large changes and Delta filtering for enhancing specific patterns of changes.

Moreover, an edge-based change detection algorithm was developed in [34], which requires a single target region and a reference region to be supplied. A target region is an area that represents the interested area, in which a change is to be detected. A reference region is an area that represents a homogeneous textural area where no change occurs. Since the edge-based change detection algorithm depends heavily on accurate reference region information, each target must have an appropriate reference region. Ideally, the reference region covers an area that is identical to the area of the target region when no change has occurred.

No matter what the change detection algorithm is, performing the change detection requires the same data scale or model. By surveying the whole research field for the image change detection, it can be found that most of the leading image change detection algorithms are based on the raster data.

In the real world, the system may have two different data formats such as vector and raster. Some hybrid methods for change detection have been investigated. In [34] a customized Land Change Detection Tool was developed by integrating raster and vector data.

A number of comparative studies show that there is no universally optimal change detection method, the choice is dependent upon the application. However, the integrated methods provide us with a good idea to design a new approach for image change detection.

2.6 Change Detection Consideration

Although the image change detection techniques based on comparison, such as image deviation, differencing, and so on, appear to be quite rich, they are not suitable for real-time detection since much larger time series data are required. And the classification techniques for the image change detection require exact matching different categories. Considered the raster model, two important change detection techniques based on the statistical theory are mainly concerned in this dissertation.

- *Correlation Analysis*

The simple correlation analysis can be used to detect changes in a certain range. There is a high correlation between image data for regions that have not changed significantly, and a relatively low correlation between regions that have changed substantially. Correlation analysis generally fails to detect a consistent change. Especially if images are acquired under different illumination conditions.

- *Principal Components Analysis (PCA)*

As indicated in the previous method, correlation coefficient is not sufficiently stable with respect to some minor changes in the intensity value of the pixels. To resolve this problem, a method of PCA was proposed [61].

Principal components analysis is a powerful technique for extracting a structure from a potentially high-dimensional data set. It has in the recent years found

a number of applications in the fields of computer vision and pattern recognition. PCA is based on representing typical images in terms of a compact set of orthogonal basis images. The main ideas behind PCA are:

A principal component P is a linear combination of the observed features:

$$P = \sum_{i=1}^d W_i X_i, \quad (2.1)$$

where X_i is the i th feature and the weight W_i is chosen to maximize the ratio of the variance of P to the total variation, subject to the constraint

$$\sum_{i=1}^d W_i^2 = 1. \quad (2.2)$$

The principal components are obtained by computing the eigenvectors of the variance-covariance matrix of the features. The importance of each eigenvector is reflected by the associated eigenvalue. Each instance feature vector can be represented in terms of a linear combination of principal components.

A kernel principal component analysis was recently proposed as a nonlinear extension of a PCA. The basic idea is to first map the input space into a feature space via a nonlinear mapping and then compute the principal components in that feature space [27]. A kernel PCA has already been shown to provide a better performance than a linear PCA in several applications [53].

However, a frequent question is how many principal components are deemed adequate for a particular situation. Different criteria have been proposed. Some criteria set a threshold for eigenvalues so that principal components with associated eigenvalues less than this threshold are deleted. Some criteria set a threshold for the variation accounted for and select the principal components with larger eigenvalues first until the threshold is reached. Sometimes a fixed number of principal components with the largest eigenvalues is mandatorily selected [17].

From real-time or near real-time point of view, it is no doubt that the correlation analysis should be an optimal choice.

Chapter 3

A FLEXIBLE CONFLATION MODEL

Although many efforts have been made to solve conflation problems, it is still a challenging research field because of the complexity of real applications. More specifically, it appears that a satisfactory level of reliability has not been achieved from the real-time or near real-time point of view. Therefore, the further attention towards enhancing the conflation ability will be stressed.

Assume that the data on hand is in vector format such as Vector Product Format (VPF), and the latest or new data comes from satellite image, the study endeavor of the dissertation is to explore the ways in which the conflation accuracy can be obtained with the aid of change detection techniques. The perspective regarding vector conflation comes from the fact that many vector products in GIS already exist and are in field use. With vector databases, the dissertation will focus on vector-vector conflation problems.

3.1 Data Models in GIS and Their Conversion

There are two main models for the representation of data in a GIS, that is, the raster model and the vector model [28]. These models are the basis of conflation.

The raster data model is based on an array or grid of square cells, each cell represents a square parcel of the real world. A value is assigned to each cell, which represents an attribute of the real world parcel. Because of its data structure, the raster model is the best format for the applications such as slope, land cover types, remote sensing, satellite imagery, aerial photographs, and so on.

The vector data model represents objects as points, lines, and polygons that are referenced to real world objects using coordinate systems. The vector model allows you to store topological information. The primary limitation of the vector model is the inability to represent, analyze, and process continuous data such as aerial photographs, and satellite images.

Figure 3.1 shows the real image based on raster model. Figure 3.1(a) presents Morphological Raster Data which includes building, open area, water and hard surface. Figure 3.1(b) provides the building outlines represented as raster data and polygons, respectively. The difference between raster and vector representation of a building outline is obviously shown in Figure 3.1(c).

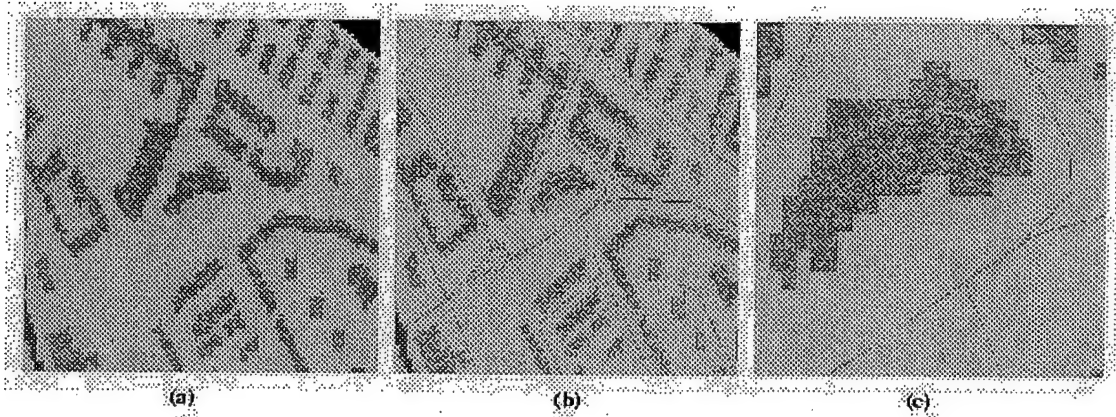


Figure 3.1: Examples of Raster and Vector Data

In general, the vector data structure produces smaller file sizes than raster image because a raster image needs space for all pixels while only point coordinates are stored in the vector representation. Besides the size issue, the vector is easier than raster data to handle because it has fewer data items and it is more flexible to be adjusted for a different scale. Although the vector data structure is the choice as the primary form for handling graphical data in most GISs, the vector data acquisition is often more difficult than the raster image acquisition due to its abstract data structure, and topology between objects and attributes associated.

The data conversion refers to the process of converting data from one format to another. In the real world, such a conversion is necessary due to the suitability of different data formats for different purposes. For example,

- The data on hand is in a vector format, but latest update comes from the satellite image format (raster). Such applications include creating a vegetation map from classified satellite data [56].
- One may use statistical tools to analyze the information based on a raster format, but the data in hand is in vector format. This requires converting a vector dataset

to a raster image. A typical application is to compare the satellite image with census data.

Today many vendors have provided the tools for conversion between the two different models. Therefore, it is not difficult to deal with both vector data and raster data in real applications.

3.2 A Mobile Agent Prototype Model for Conflation

With a growing abundance of geo-spatial data, the situations where more than one data set is available to meet a particular need are becoming common[20]. In order to create a best possible database, the data conflation is needed. For performing data conflation, a general distributed mobile agent model can be designed as Figure 3.2.

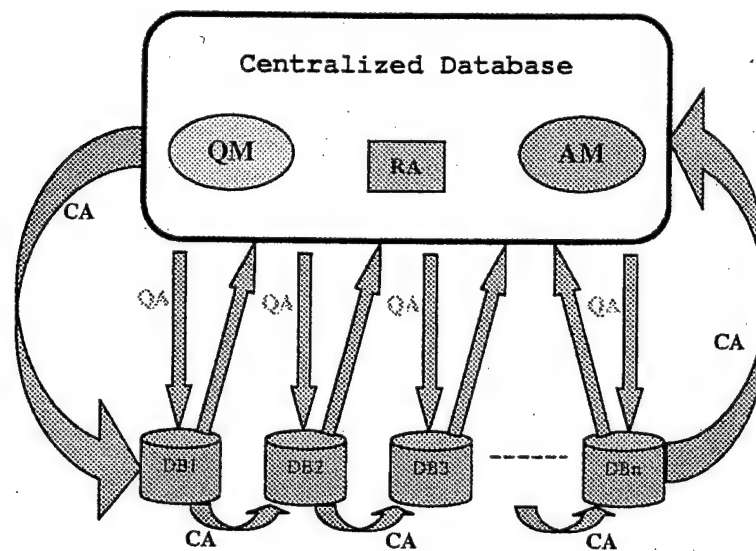


Figure 3.2: A general distributed mobile agent model.

The architecture consists of a primary, centralized database and multiple, heterogeneous databases. Based on this architecture, there exist multiple agent classes. A brief description is given as follows:

- QM is a queue manager: It is responsible for supervising a priority queue generated by the ROI agents. Where ROI stands for the Region Of Interest.

- AM is an agent manager: Its responsibility is to corporate agent classes.
- RA represents ROI Agent: Each RA is responsible for managing updates for a particular Region of Interest. RAs are static, remaining on the centralized database.
- CA represents Conflation Agent: The CA is a super class of many specialized agent classes that have extensive knowledge about their domain relevant to the conflation process. Moreover, CA is the intelligent mobile agent which travels to the heterogeneous databases to perform conflation.
- QA represents Querying Agent: It is released by the centralized database to gather information for general or conflation-related queries. The generator or creator of QAs varies with the different purposes.

A typical scenario of performing conflation can be described as a following process:

Given a specific task, the information related to the specific task might be stored in multiple databases. The system will generate and launch QAs to collect data from sources. When the customized and intelligent CAs travel to the sources, they bring the data set collected by QA to the matching knowledge base where a conflation algorithm will detect changes and select the best information.

Since the conflation algorithms often deal with complicated problems in real world, the conflation process is regarded as a practical exercise. Due to the development of Internet/Intranet and Open GIS, there are great opportunities to get data from different data providers. One of the promising applications in GIS is to integrate data from different sources. However, how to extract useful information and combine this information to meet the specific requirements is still a great challenge. Some research results show that a potential solution of GIS is to use an integrated hybrid of advanced technologies (i.e., artificial neural networks, fuzzy logic, evidential reasoning, and so on) to help automatically integrate, reconcile, and update multiple disparate geo-spatial databases in near real time.

In confronting a distributed environment shown in Figure 3.2, by taking the potential advantages of artificial intelligent technologies, a general conceptual model for conflation can be shown as Figure 3.3.

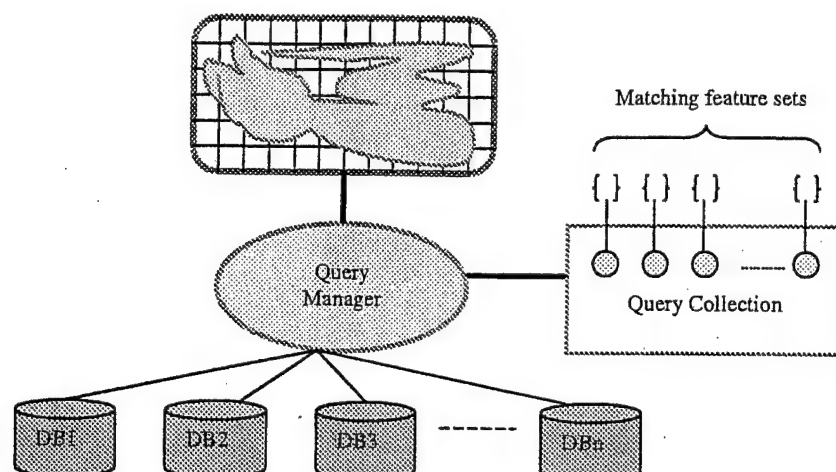


Figure 3.3: A conceptual model for distributed conflation

The process can be briefly described as follows. The query manager retrieves all feature objects from the distributed databases that store the related information, and collects the objects. Any object determined to be a match is placed in the matching feature set for conflating, and ranked according to similarity scores. The Conflation agent launched from the central database will travel to the matching feature set to perform the conflation process.

3.3 A Raster-Based Vector Conflation Model

From the previous work, the following consideration or knowledge will be very helpful for designing a conflation scheme.

- The conflation is most commonly done with the vector map.
- Most of the leading algorithms for image change detection are based on the raster data.
- The conflation of vector data, itself, is generally a costly and time-consuming task involving feature-matching between two data sets and then transferring the feature's attributes from one data set to the other.
- Since performing conflation or change detection requires the same data scale or model, it is unavoidable to undertake the transformation process during conflation and change detection. However, the transformation between vector and raster is

not a short time process. Especially, the vectorization, to generate the vector data from original image, is expensive and time consuming.

Based on the above considerations, a raster-based vector conflation model is developed as shown in Figure 3.4. The principal purpose is to use a vector version conflation algorithm and raster-based change detection. A general scheme of this conflation model includes the following steps:

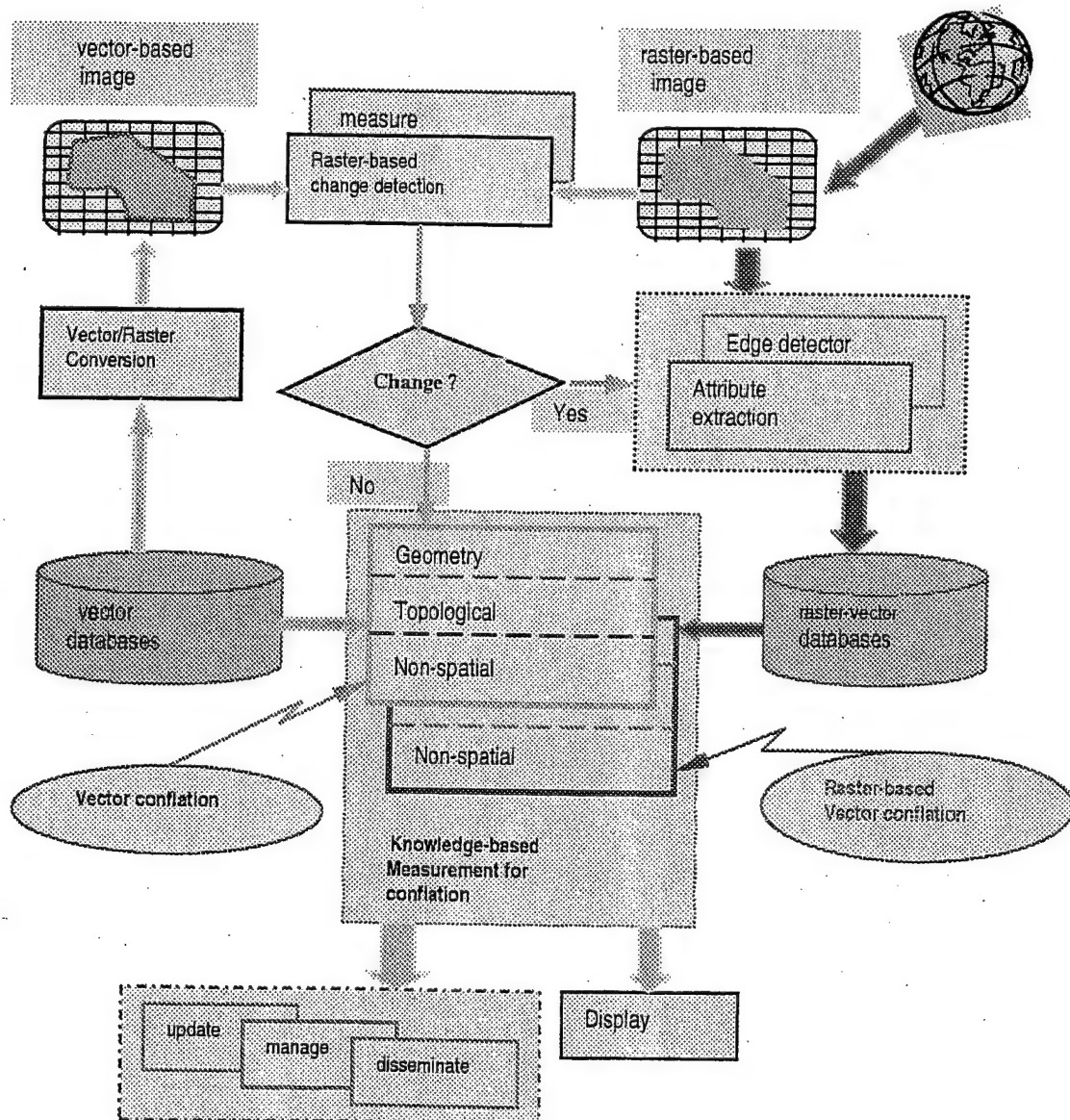


Figure 3.4: A raster-based vector conflation model

- Design knowledge bases for conflation based on the vector data;
- Design knowledge bases for change detection by means of the raster data coming from near real-time satellite image;
- Transform vector data to the raster image;
- Develop vector version conflation algorithm;
- Develop an algorithm for change detection which will detect the information change significantly; and,
- Vectorize raster data - this process will involve finding a set of pixels that can approximate points, lines, and polygons of the vector data.

The final step is to select one of the conflation algorithms according to the results of the raster-based image change detection. If no change occurs, it is a regular conflation process. Otherwise, it will perform the raster-based vector conflation.

Each of the above steps in the scheme requires a sufficient study in order to implement an appropriate and practical conflation process. However, the focus of this dissertation will be placed specifically on processing:

- The conflation such as spatial and non-spatial conflation based on VPF databases.
- Image change detection in terms of satellite images.

Moreover, since the uncertainty in geospatial data is an ancient problem, it has been the subject of a growing volume of research during the past decade and continues to figure prominently in research agendas[20]. The dissertation will put special efforts on dealing with uncertainty.

3.4 Significance of the New Model

The scheme of the raster-based vector model in Figure 3.4 is developed for the vector data format, and can handle changes over time by means of raster data. It can work in the different ways such as:

- **Pre-detection/Post-conflation:**

In this way the image change detection can be done at each vector database before conflation. This is a case in which every database can have the satellite image information for updating.

- **Pre-conflation/Post-detection:**

In this way regular conflation goes first. After conflation among the multiple vector databases, the "best" one will be selected to perform image the change detection with the raster image. If no any changes occur, the scheme works in the exact same way as before shown in Figure 3.3. Otherwise, it performs a raster-based vector conflation again.

Therefore, the new model is a more general and flexible conflation model.

Chapter 4

A VECTOR-BASED CONFLATION SCHEME

Geographic datasets that have both rich attribution and good positional accuracy can be developed through conflation. There exist quite extensive literatures on conflation. Most of these methods employ statistical tools, feature matching and neural-nets. Each of these techniques has its own merits and demerits. One generic limitation of these works lies in exact matching of spatial features/attributes.

Conflation is such a large topic that mastery can only be achieved through experience. Fuzzy logic, which has been proven to be successful in inexact environment, can equally be used for inexact matching in conflation problems. Recently, some approaches using AI techniques such as fuzzy membership function have been developed [12, 63]. The desirable conflation algorithm mentioned before, which was proposed in [7, 8], utilized a hierarchical rule-based approach for feature matching. But none of these techniques consider both AI and statistical techniques. Therefore, the current conflation algorithms may work reasonably well on one type of conflation problems, there is no general method.

The premise of this dissertation has been that conflation is an inexact process. The geodata sets in conflation are defined as digital spatial files such as VPF files that cover the same area, describe the same information and may vary in density and accuracy. In such a dataset, three types of geographic objects, i.e. points, lines, and polygons, are used to simplify and symbolize the complex real world, and the attribute information associated with objects gives meaning of objects and distinguishes objects from one another. The purposes of conflation include 1). increasing spatial accuracy and consistency; 2). adding new spatial features into datasets; 3). updating and/or adding more attributes. The intention is to develop a conflation scheme by which components can be integrated to solve any specific conflation problems.

4.1 Conflation Components

A number of real world problems are complicated. Because of different representations of the real world, such as different scales, different data standards, different classifications and different semantic meanings, some matching criteria are good for

one case, but for other cases. It is difficult to use a single algorithm for dealing with all conflation problems. The component-ware technology provides a way to solve this problem.

In contrast with the traditional database applications, GIS applications require both spatial and non-spatial data. Within vector databases or datasets, there are basic types of information for features, that is, geometric properties (location), topological properties (relationships), and non-spatial properties (attributes). The Vector Product Format (VPF) specifies a georelational framework based on a vector data model that is suitable for large geographic database. In such a large database, inconsistencies may be caused in all geometric, topological and attribute aspects. It is reasonable to consider conflation in each aspect. By means of the component-ware technology, which is a method that can be used to develop independent but interoperable components, conflation components in a vector-based database can be designed as following. It is desirable to assemble these components to solve any specific problems.

4.1.1 Geometric Conflation Component

Geometry helps us to understand the representation of the spatial position including its shape and size. With vector GIS, geographic objects are represented geometrically by points, lines, or polygons [5].

Generally, the need for geometric conflation on images arises from the fact that we want to "force" the object coordinates to fit a designated coordinates. This is required to be able to measure the position, size, distance, and other geometric parameters of the objects.

- **Points:** The objects that occupy very little or no arial extent are often represented as *points*. Thus, a point is a 0-dimensional geometric primitive. The spatial position of a point is described by one set of coordinates referring to one georeference system.
- **Lines:** A line is a bounded continuous 1-dimensional geometric primitive. Linear features are best described as *lines*.
- **Areas or regions:** An area is a bounded continuous 2-dimensional geometric primitive. A boundary is a closed 1-dimensional non-intersecting element defined by a boundary line. Within a vector-based database, areas or regions are usually structured as *polygons*[32].

As a whole, points, lines, and polygons are referred as to geographic objects since they represent geographic features. Such a conflation component will be used to deal with spatial object inconsistencies.

4.1.2 Attribute Conflation Component

Metadata contain a wide range of information about the image data to assist users in determining the availability, quality, and usefulness of the data. Attribute information is one of major information in the metadata.

Attributes provide meanings to the geographic features. For example, the color of a building, the width of a road and so on. There are four types of attributes: [5]

- Quantitative (also called continuous) attributes, which correspond to quantities that can be measured in a given unit. Examples include width, temperature, height;
- Qualitative (also called discrete) attributes, which do not correspond to quantities but usually to a finite set of values that can be enumerated. Examples include classes, codes;
- Geometry; and,
- Description, which includes text, graphs, photographs.

The component of attribute conflation will be used to match features. The attribute conflation algorithms are also referred to as the semantic methods. They can be used to match features very efficiently if both datasets defined a common attribute field and the semantics of both data sets are known. Little research has been done in this aspect. Usually, it will be implemented at the metadata level.

4.1.3 Topological Conflation Component

Topology is the study of the characteristics of geometrical objects that are independent of the underlying coordinate system. Usually, topological relationships express the concepts of inclusion and neighborhood. The main purpose for providing topological information in GIS is to improve spatial analysis capabilities.

Currently, VPF supports four levels topology:

- Level 0 — boundary representation only; no intersections of lines or areas are considered.
- Level 1 — non-planar graph, commonly referred to as “spaghetti”; it is suitable for representing networks.
- Level 2 — planar graph, in which no edges overlap.
- Level 3 — full topology, in which no faces overlap.

With respect to full topology, topological conflation component will be used to search range for attribute conflation, and check the results of geometric matching. It requires available topological information such as connectivity, and adjacency. They are seldom used alone. Sometimes they are aided by a geometrical method.

4.2 Intelligent Conflation Algorithms

Knowledge plays a critical role in an intelligent system. The analysis of geographic data requires a large body of knowledge about geographic properties. Therefore, more attention should be paid to knowledge structure and knowledge processing.

Within VPF databases, conflation will be performed on two coverage. The “coverage” is a key term that refers to the set of all geospatial features for which topological relationships have been established within a specified geographic area. Arc/Info defines a coverage as[15]:

“It (a coverage) generally represents a single set of geographic objects such as roads, parcels, soil units, or forest stands in a given area. A coverage supports the georelational model – it contains both the spatial (location) and attribute (descriptive) data for geographic features.”

By conflation process, differences in the features’ geographic locations and attribute values are reconciled. Features that don’t have corresponding features in the other source are identified and can be added. In this section, extraction of knowledge from the coverage is provided, corresponding conflation algorithms are discussed.

4.2.1 Conflation Algorithm for Topology

Precise vertex-to-vertex and line-to-line matching were just not possible. The topological conflation algorithm requires an available topology. Within the VPF databases, the "winged-edge" format is one of several ways for representing full topology, and is the one specified for level-3 coverages. A topological construct of the "winged-edge" is that each edge is connected to two of its neighboring edges, a neighboring edge is any edge that shares a start or end node with the original edge. Here, an edge has a start node, which is connected to the left edge, and an end node, which is connected to the right edge. Thus, the topological structure at a particular area can be regarded as a network topology which can be represented by linguistic variables. Since a nature description of the network topology, such as round shape, large size and so on, is much better than numerical or formula expression, this representation is more in agreement with the way a person might describe some of these attributes.

Based on the idea that uses the linguistic variables to interpret the characteristics of the topology, it is possible to design an intelligent conflation algorithm by means of fuzzy logical and inexact reasoning methods. However, when topology is not available in the source datasets, it is necessary to build a topology.

4.2.2 Conflation Algorithm for Geometry

Geometric distortions commonly occur in source data due to imperfect registration, lack of geodetic control, and a variety of other causes. Rubber sheeting will be used to correct flaws through the geometric adjustment of coordinates.

Assume that one of the coverages between coverages is identified to be more spatially accurate. This coverage is referred to as the reference coverage. The geometry of the reference coverage is not modified during the conflation process. The other, less spatially accurate coverage is transformed via rubber-sheeting to match the reference geometry during conflation.

With vector GIS, geographic features are represented geometrically by points, lines, and/or polygons in a two-dimensional space. For these features, there exist different matching types such as point-to-point, point-to-line, line-to-line matching, etc. shown as in Figure 4.1.

Generally, different conflation approaches use different criteria for matching processing. The common geometric criteria can be briefly described as following:

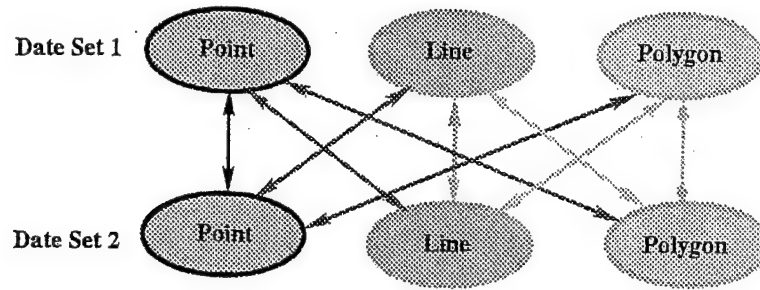


Figure 4.1: Spatial Feature Matching Types

Euclidean distance: used to calculate the distance from point to point, point to line, etc.

Given two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, a mathematical expression for Euclidean distance D can be simply given as

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \quad (4.1)$$

Hausdorff distance: used to calculate distance between two linear features to search for line-to-line matches.

$$DH = \max(d_1, d_2). \quad (4.2)$$

Where, d_1 denotes the largest minimum distance from line 1 to line 2 and d_2 is the largest minimum distance from line 2 to line 1.

Fourier descriptor: used for polygon shape feature.

Fourier descriptors are a series of Fourier coefficients that define the polygon boundary.

If several matches are found by distance criteria, additional criteria such as angular criteria may be needed. However, the geometric conflation method requires that two datasets have similarity in geometric location. If necessary, rubber sheeting may be involved for position alignment.

The geometry conflation is carried out by performing the following matching:

1. searching for node pairs between two coverages.

2. matched node pairs are used to generate a rubber-sheeting transformation that brings two coverages into better alignment with the reference coverage.

Note: Rubber-sheeting and node matching proceed iteratively. Each iteration produces a new transformation which brings the coverages into better alignment possibly.

3. Line matching proceeds once node matching has been completed.
4. The next step is feature merging. The features that have no corresponding feature in the other source will be merged.

Where, node matching is performed to create rubber-sheeting transformations. Distance measures are used for matching nodes.

- *Point to Point*: Euclidean distance matching.
- *Point to Line*: Point to line distance.
- *Point to Polygon*: Point in polygon.
- *Line to Line*: Hausdorff distance matching.
- *Line to Polygon*: Line to polygon matching can be converted into line to polygon boundary matching.
- *Polygon to Polygon*: Polygon centroid, point in polygon, polygon shape feature.

In general, the conflation match results will be more accurate when two sources are similar.

4.2.3 An Intelligent Algorithm for Attribute Conflation

The attributes associated with features help to describe their unique characteristics. The attributes of spatial features may have something in common but have different semantic definitions. In VPF, attribute table will collect the identically defined attribute rows [40]. An example of attribute table is shown in Table 4.1.

Our attention to the attribute conflation is turned on the intelligent attribute matching algorithm [7]. For attribute matching, each feature object is considered as a set of attribute-value pairs, that is:

Table 4.1: State Attribute Table

ID	State	Area(sq. mi.)	Total Population
implicit	character string	binary integer	binary integer
UNIQUE-KEY	PRIMARY-KEY	NON-UNIQUE	NON-UNIQUE
1	California	158706	26365000
2	New York	110561	936000
3	Utah	84899	1645000

$$\{ (a_{11}, v_{11}), (a_{12}, v_{12}), \dots, (a_{1n}, v_{1n}) \}$$

$$\{ (a_{21}, v_{21}), (a_{22}, v_{22}), \dots, (a_{2m}, v_{2m}) \}$$

Where the two categories of numeric and linguistic attribute domains are considered for matching. In general, matching for numeric domains is handled through the use of membership matching functions, while matching for linguistic domains is handled through the use of attribute similarity tables. For an example, the linguistic domain is discussed as follows.

A similarity table for a specific attribute contains a value in the range $[0, 1]$ for each attribute domain value. Each of these values represents a degree of matching between two attribute values. In many cases, the domain values are integers that represent encodings of linguistic characteristics; thus, the similarity values in the table represent similarity between linguistic terms. Matching for features based on attribute similarity is a two-phase process.

- First, the similarity between each of the attribute values for two features is determined from a similarity table.
- Second, measures of semantic interrelationships between and among the various attribute values are computed within a rule-based expert system. Based on these interrelationships, the expert system returns one or more weights for increasing or decreasing the matching score for various attributes.

Chapter 5

IMAGE CHANGE DETECTION

The integration of image data into GIS is one of the great ideas whose time has come. As an active research field in GIS, image change detection is considered as a potential application of the presently available, high-resolution imagery from commercial earth observation satellite. This study will exploit remotely sensed images of the same scene for image change detection.

Although the image change detection techniques based on the statistical theory have been applied in many applications, they do not yield much useful information in the nature of the change. Because of uncertainty in the image processing, it is more difficult to make the interpretation of the changes. Normally it indicates that a statistically significant change has occurred somewhere in the region under examination. Therefore, the results of the change analysis can be very complex and unclear at a glance.

In the recent years, some potential techniques for the new change detection emerge from the areas such as computer vision, image understanding[36], knowledge-based systems, and fuzzy set theory. Fuzzy logic is a way of thinking that seems particularly appropriate for this purpose. Some studies reported that a rule-based fuzzy logic approach gave better results than a maximum likelihood classifier. The new techniques offer us some promise to develop new algorithms for the image change detection.

By surveying current detection methods, most previous work dealt with the use of just one change detection technique which is applied to some particular problems in a particular area. In this chapter, a hybrid method, which takes the advantage of the statistic analysis tools and fuzzy theory, is investigated in order to deal with a real-time image change. The main idea is the two-step change detection, that is, a pre-detection technique (such as correlation analysis, histogram analysis) determines which of the changes are significant, and the post-detection technique provides more information about how these changes can be trusted.

As we know, to understand the image in human is the result of reasoning in terms of knowledge. This knowledge necessarily includes uncertainty and fuzziness, which reflect the incompleteness and imprecision inherent in any human knowledge of the real world.

The main effort of the dissertation is to investigate an inferencing approach which can perform fuzzy inference under uncertainty. By introducing a Certainty Factor (CF), a hierarchical inferencing structure is proposed. Also, the basic concepts related to raster images and fuzzy sets are provided in this chapter.

5.1 Basic Concepts for Raster Data

The raster model is one of the major families of representation models for image. It divides the region into rectangular building blocks (grid cell or pixels) that are filled with the measured attribute values.

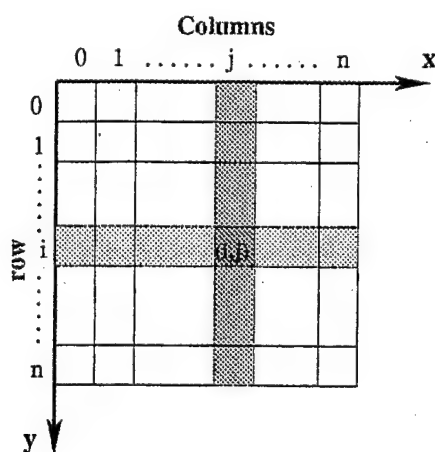


Figure 5.1: Discrete image I .

Figure 5.1 shows a discrete $n \times n$ image I in the raster format. The position of pixel (i, j) is given in the common notation for matrices. The first index, i , denotes the position of the row; the second, j , the position of the column [24]. I_{ij} represents the brightness of the image in the corresponding cell (i, j) .

Pixels are elementary units. As long as the statistical properties of a pixel do not depend on its neighbor pixels, the classical concepts of statistics can be applied. Based on the raster model, we consider a 8-bit gray image that has been partitioned into n^2 non-overlapped blocks of equal dimensions. Some measure of the distribution of the pixel values in an image, and a set of image features such as edge, shade and mixed-range can be defined as follows.

5.1.1 Common Statistic Measures

An image can be regarded as a function $I : R^n \mapsto R^m$, where normally n is 2 and m is 1 for intensity or 3 for color. If $m = 1$, the value I_{ij} is called the gray level of pixel (i,j) . If $m > 1$, I_{ij} is referred to a feature vector. In this study, the gray values will be considered.

Considering that a pixel is a random variable as any other measured quantity, first-order statistics such as mean values, and the variance, or standard deviation [38, 39] can be defined as follows.

Definition 5.1.1. The *Mean* of the pixel values in an image, I_{avg} , is given by

$$I_{avg} = \frac{1}{n^2} \sum_{i,j} I_{ij}. \quad (5.1)$$

Definition 5.1.2. The *Standard deviation* of the pixel values in an image, σ^2 , is given by

$$\sigma^2 = \frac{1}{n^2} \sum_{i,j} (I_{ij} - I_{avg})^2. \quad (5.2)$$

The standard deviation, or more generally, the second moment, is one measure for the degree of smoothing. This measure can be applied in the spatial domain.

Definition 5.1.3. *Correlation R.* Given a pair of images of the same area acquired at different times. Correlation starts by computing the correlation coefficient for a local neighborhood of $n \times n$ pixels. The value of correlation coefficient R can be calculated as:

$$R = \sum_{i=1}^{n^2} \sum_{j=1}^{n^2} \frac{(a_{i,j} - a_{avg})(b_{i,j} - b_{avg})}{n^2 \sigma_a \sigma_b}, \quad (5.3)$$

where, $a_{*,*}$ and $b_{*,*}$ are pixel intensity values in the first and second images; a_{avg} , σ_a , b_{avg} , σ_b are mean and standard deviation of the intensity, respectively.

Definition 5.1.4. *Image histogram, $h(I_{ij})$.* A histogram of an image is a list which contains as many elements as quantization levels. In each element, the number of pixels is stored that show the corresponding gray value. The domain of the histogram is the set of possible pixel gray values (quantization levels).

Individual image data are typically quantized with brightness values ranging from 2^8 to 2^{12} . The majority of current data are quantized as 8-bits, with values ranging from 0 to 255 [25].

Thus, a histogram can be also defined as a frequency distribution graph of a set of numbers, that is a graph with “pixel value (or brightness)” on the horizontal axis from 0 to 255 and the “number of pixels” on the vertical axis.

Tabulating the frequency of occurrence of each brightness value within the image provides the statistical information that can be displayed graphically in a histogram. Figure 5.2 shows an example to calculate and plot the histogram of an image.

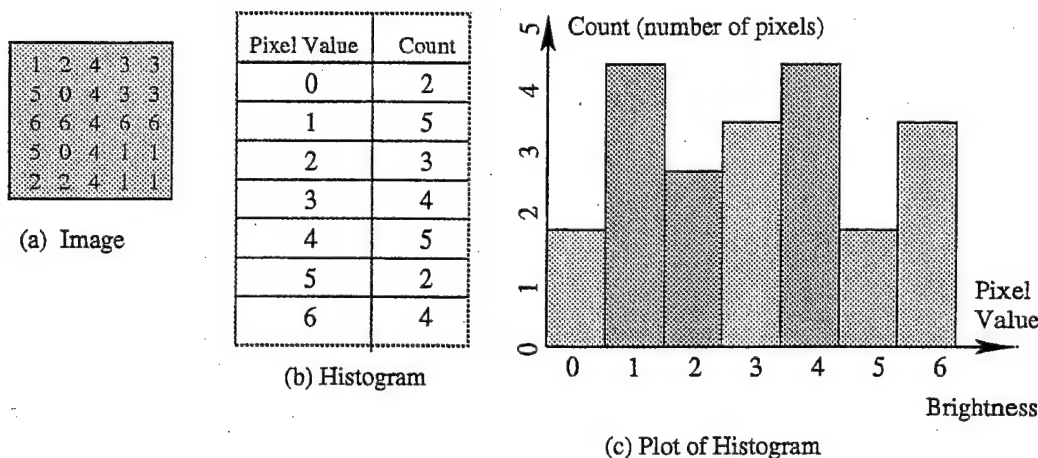


Figure 5.2: An 1-D histogram

5.1.2 Image Features

The extraction of Image feature requires neighborhood operators that are sensitive to changes of constant gray values. The base of image feature extracting is differentiation. In discrete images, differentiation has to be approximated by differences between neighboring pixels.

Definition 5.1.5. An *edge* is a contour of pixels of large gradient with respect to its neighbors in an image.

Definition 5.1.6. A *shade* is a region over an image with a small or no variation of gray levels.

Definition 5.1.7. A *mixed range* is a region excluding edges and shades on a given image.

Definition 5.1.8. The *gradient* [46] at a pixel (i, j) in an image is estimated by taking the square root of the sum of difference of gray levels of neighboring pixels with respect to the pixel (i, j) .

$$G_{ij}^2 = \frac{1}{2} \left(\sum_{k=i-1}^{i+1} (G_{kj} - G_{ij}) \right)^2 + \frac{1}{2} \left(\sum_{k=j-1}^{j+1} (G_{ik} - G_{ij}) \right)^2. \quad (5.4)$$

Definition 5.1.9. The *gradient median* G_{medi} within a partitioned block is defined as the middle gradient values in that block when all gradient values are arranged in ascending or descending order. It is another measure of central tendency.

Definition 5.1.10. The *gradient average* G_{avg} within a block is defined as the average of the gradient of all pixels within that block,

$$G_{avg} = \sum_{i,j} G(i, j) \cdot p(G_{ij}), \quad (5.5)$$

where G_{ij} denotes the gradient values at pixels, and $P(G_{ij})$ represents the probability of the particular gradient G_{ij} in that block [62].

Definition 5.1.11. The *variance* σ^2 of the *gradient* is defined as the arithmetic mean of square of deviation from mean. It is expressed formally as,

$$\sigma^2 = \sum_{i,j} (G_{ij} - G_{avg})^2 \cdot P(G_{ij}). \quad (5.6)$$

5.2 Basic Concepts of Fuzzy sets

To deal with non-exact problems is a wider part of human experience. In 1965, Zadeh introduced the idea 'fuzzy sets' to deal with inexact concepts in a definable way. Since 1960s, the theory of fuzzy sets has been developed to the point where useful, practical tools are available for use in other disciplines. Fuzziness is often a concomitant of complexity. It is appropriate to use fuzzy sets whenever we have to deal with ambiguity, vagueness and ambivalence in mathematical or conceptual models of empirical phenomena.

5.2.1 Fuzzy Sets

If X is a collection of objects denoted generally by x , then a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) | x \in X\}, \quad (5.7)$$

where $\mu_A(x)$ is called the membership function (or MF for short) for the fuzzy set A . It denotes the degree of membership of a variable x to belong to A , where A is a subset of a universal set X . Usually, X is referred to as the universe of discourse, or simply the universe.

5.2.2 Membership functions

A fuzzy set is completely characterized by its membership function (MF). Put simply, in fuzzy sets, the grade of membership is expressed in terms of a scale that can vary continuously between 0 and 1. A more convenient and concise way to define the MF is to express it as a mathematical formula.

Due to the smoothness and concise notation, Gaussian function is becoming increasingly popular for specifying a fuzzy set. Moreover, Gaussian functions are well-known in probability and statistics, and they possess useful properties such as invariance under multiplication and Fourier transform.

Definition 5.2.1. A Gaussian MF is specified by two parameters, $\{c, \sigma\}$, i.e.

$$Gaussian(x; c, \sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2}, \quad (5.8)$$

where c represents the MFs center and σ determines the MFs width.

Figure 5.3 plots a Gaussian MF defined by $Gaussian(x; 5, 2)$.

5.2.3 Set-Theoretic Operations

Union and intersection are the most basic operations on classical sets. Corresponding to the ordinary set operations of union and intersection, fuzzy sets have similar operations, which were initially defined in Zadeh's seminal paper[29].

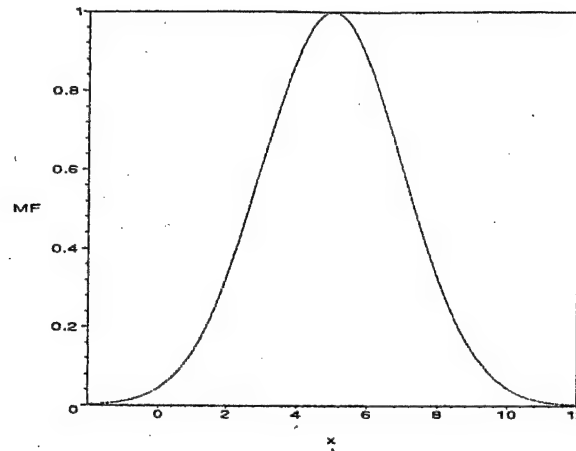


Figure 5.3: Gaussian membership function

Definition 5.2.2. Union (disjunction)

The union of two fuzzy sets A and B is a fuzzy set C , written as $C = A \cup B$ or $C = A$ OR B , whose MF is related to those of A and B by

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)). \quad (5.9)$$

Definition 5.2.3. Intersection (conjunction)

The intersection of two fuzzy sets A and B is a fuzzy set C , written as $C = A \cap B$ or $C = A$ AND B , whose MF is related to those of A and B by

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)). \quad (5.10)$$

5.2.4 Linguistic Variables and Other Related Terminology

Knowledge in the fuzzy model is represented by a linguistic variable, which is characterized by fuzzy sets. A linguistic variable is characterized by a quintuple [23],

$$(x, T(x), X, NR, MR),$$

where

x is the name of the variable;

$T(x)$ is the term set of x , that is, the set of its linguistic values or linguistic terms;

X is the universe of discourse;

NR is a syntactic rule which generates the terms in $T(x)$; and,

MR is a semantic rule which associates with each linguistic value.

For example, an image can be interpreted as a linguistic variable, then its term set $T(image)$ could be expressed as,

$$T(image) = \{\text{edge, shade, mixed range}\},$$

where each term in $T(image)$ is characterized by a fuzzy set of a universe of discourse X .

Usually we use "Image includes edges" to denote the assignment of the linguistic value "edge" to the linguistic variable "image". The syntactic rule refers to the way the linguistic values in the term set $T(image)$ are generated. The semantic rule defines the membership function of each linguistic value of the term set.

5.3 A Hybrid Approach for Image Change Detection

It is true that each detection technique has its own merits and demerits. For practical applications, a reliable approach depends on both of the measured quantity and its capability to handle uncertainty. The research is dedicated to develop a more flexible and reliable method for image change detection based on satellite images. A two-step processing method is designed, i.e. pre-detection and post-detection.

The pre-detection is a process in which detecting changes can be analyzed from the standpoint of statistical decision theory. And the post-detection is a process which adopts fuzzy set theory to make a decision.

5.3.1 Pre-detection – A Statistic Analysis

From the statistic point of view, some basic characteristics for change detection techniques can be described as follows:

- As usual, the mean of a distribution is used as a measure of the distribution's location. The estimate of the mean gray value of the region, averaging within a local neighborhood, appears to be a central tool for region detection. However, the mean is a poor measure of central tendency when the set of observations is skewed or contains an extreme value.
- The standard deviation is a measure of the absolute dispersion. A small standard deviation suggests that pixel values are clustered tightly around a central value.

demonstrates there is no relationship between two sets. Therefore, there is a high correlation between image data in the regions that have not changed significantly, and a relatively low correlation between regions that have changed substantially. However, correlation analysis generally fails to detect a consistent change. Especially if images are acquired under different illumination conditions.

3. Gray Scale Normalization

Changes in the intensity at pixels occur for many reasons such as noise, illumination variations. It is often necessary to normalize the gray scale of an image in order to improve the reliability of feature detection or measurement. For instance, suppose that we want to compare two images I_1 and I_2 in order to detect differences between them. If the images were obtained under different lighting conditions, we must somehow compensate for this, since otherwise they will have different gray levels at every point even if they are images of the same scene. A common method of gray-scale normalization is to standardize the image histogram, i.e. the gray level frequency distribution. By normalization, an image is brought into a standard form.

Consider two images I_1 and I_2 , having n^2 points. Suppose that I_1 is quantized to k levels l_1, l_2, \dots, l_k , I_2 is quantized to k levels $\hat{l}_1, \hat{l}_2, \dots, \hat{l}_k$, and level l_i and \hat{l}_i have m_i points. Then, if the illumination variations, and/ or white noise is considered, the following relationship should be satisfied:

Since

$$\bar{I}_1 = \frac{1}{n^2} \sum_{i=1}^k l_i m_i, \quad (5.14)$$

$$\bar{I}_2 = \frac{1}{n^2} \sum_{i=1}^k \hat{l}_i m_i. \quad (5.15)$$

Let

$$\hat{l}_i = l_i + \alpha, \quad (5.16)$$

where α is a constant.

Then,

$$\bar{I}_2 = \bar{I}_1 + \alpha, \quad (5.17)$$

$$\sigma_1 = \sigma_2. \quad (5.18)$$

The normalization of histogram \hat{h}_2 of image I_2 can be computed by:

$$\hat{h}_2 = h_2 - \alpha. \quad (5.19)$$

In this way, the normalization of the gray scale makes feature values independent of, or at least insensitive to noise and illumination.

Based on the above information, the pre-detection technique can be divided as:

Step 1: Correlation Analysis for Small Change

The correlation R is used to analyze the small change in the given area. A simple rule for judgment can be given as:

if the correlation $R \in (0.9, 1]$, no change is considered.

if the correlation $R \in (0.5, 0.9]$, some changes may happen.

Otherwise, changes occur.

Step 2: Histogram Analysis for Structure Change

Due to the poor quality of correlation analysis to detect structure change, and also gray scale under or overflow is a common error which often goes unnoticed and causes a serious bias in processing, the further step for image change detection is needed.

Since the histogram of an image can clearly illustrate contrast and multimodal in nature, that is, each element in the histogram tends to be comprised of gray levels different from one another, the histogram can be used as a useful and important graphical aid to understand the information content of a satellite image.

Firstly, the histograms of images are normalized in order to compare them. Secondly, based on their gray level composition, the number of the peaks in the histogram is counted. Since each peak corresponds to a dominant type of the objects in the image, finally, the statistic structure change can be derived by comparing the number of the peaks in two images.

Let NH_1 and NH_2 be the number of the peaks in images I_1 and I_2 , respectively. If $NH_1 \neq NH_2$, the changes may occur, especially in structure.

Like all processing of data, although the histogram of an image can provide useful information about how to detect the structure change, there is a drawback or undesirable artifact to this histogram technique. Objects in a scene may be composed of gray level regions that overlap, objects will have portions that fall in another's classification. Specially, when an unusually large number of pixels have the same brightness value, or several objects have the same brightness value, the traditional histogram display may not be the best way to represent the information content of an image. Therefore, the histogram structure change detection is basic. It works best on simple scenes with objects that have distinctly different gray scale occupancies.

5.3.2 Post-detection – An Intelligent Analysis

Since it is difficult to state exact processing for image change detection in precise mathematical terms, most of image detection algorithms are heuristic [18]. How to judge these changes and give a reliable information for image change is mainly concerned. As was pointed out by Zadeh [30], conventional techniques for system analysis are intrinsically unsuited for dealing with humanistic systems, whose behavior is strongly influenced by human judgment, perception and emotions. Even though the basic change detection can be carried out based on statistic analysis, the results can not be totally trusted. This is because geographical information involves human interpretation and knowledge, which are invariably imprecise, incomplete, or not totally reliable. For dealing with these situations, human beings are highly skilled at making decision in the uncertain environment.

Zadeh proposed an alternative approach to modeling human thinking. This approach serves to summarize information and express it in terms of fuzzy sets. Recently, amount of approaches for image processing that use fuzzy membership functions have been developed. A new methodology[4, 45] that uses fuzzy membership-distance as moment descriptors for image matching gives us a clue to design our intelligent detection approach, since it is superior to all other existing techniques for inexact reasoning. The intelligent detection is provided as follows.

1. Image as a Fuzzy Subset $T(image)$

Geographical information (including satellite data) is often imprecise, meaning that the boundaries between different phenomena are fuzzy. Exact definitions are inadequate for dealing with geographic information. Mathematically speaking, an

ideal edge is a discontinuity of the spatial gray value function. It is obvious that this is only an abstraction which often does not meet the reality. For example, there is usually a gradual interface at the edge of forests and range land, in geographic space it is difficult to define a natural boundary which is sharp and well determined. Most geographic objects seem to be an abstraction of things which have unclear, fuzzy boundaries. Therefore, the interpretation of an image should contain fuzzy definitions [43, 59]. Currently there are no available fuzzy schemes to define the fuzzy subsets for image representation. This brings us to an important consideration – to interpret the significance of an image.

The extraction of image features requires to use knowledge of image processing. Generally, the partitioned blocks in an image include either edge and shades together or mixed area. Simply, an image can be interpreted as a fuzzy variable that has the primary term set {shade, edge, mixture area}. Thus, the term set of an image can be expressed as:

$$T(image) = \{\text{edge, shade, mixed area}\}.$$

This means that each block can be identified according to three possible characteristics named 'edge', 'shade' and 'mixed area'.

2. Fuzzy Membership Functions

In order to make the change detection more sensitive to imprecise (fuzzy) nature of the real world, the problem of determining the appropriate membership function drew the attention of many researchers in the image processing field.

From the statistical point of view, we can assume that each linguistic value, such as shade, edge, mixed area, corresponds to a fuzzy set whose membership function μ is represented by a Gaussian function shown as Equation 5.8. The degree of membership of a given block to contain typical sub-classes (edges, shades and mixed areas) is measured subsequently by means of basic statistic measures, for example, average gradient, variance, and median of gradients. A more challenging problem is to determine the two parameters in the Gaussian membership function. There are no any general approaches. The definition of the grades of memberships is subjective and depends on the human interpretation.

Assume that objects are characterized by constant gray values. If the pixel shows the same gray value as its neighbors, there are good reasons to believe that it lies within a region of constant gray values. Let us consider the following cases.

Case 1: Shade Membership Function μ_{shade}

When an image only contains shades, the ideal gradient average G_{avg} should be zero. By Definitions 5.1.2, the gradient median G_{medi} and the variance of gradient σ^2 are also zero ideally. It is intuitive that the degree of membership of the image to belong to "shade" will decrease when G_{avg} increases. Based on these analysis, it is reasonable to presume the shade membership function with respect to the gradient average $\mu_{shade}(G_{avg})$ as

$$\mu_{shade}(G_{avg}) = e^{-A_{s1}G_{avg}^2}, \quad (5.20)$$

where A_{s1} is a constant, which can be determined by boundary conditions.

In the same way, the shade membership function with respect to parameter G_{medi} and σ can be realized with a similar form

$$\mu_{shade}(G_{medi}) = e^{-A_{s2}G_{medi}^2}, \quad (5.21)$$

$$\mu_{shade}(\sigma) = e^{-A_{s3}\sigma^2}, \quad (5.22)$$

where A_{s2} and A_{s3} are constant and determined by boundary conditions.

Case 2: Edge Membership Function μ_{edge}

By Definition 5.1.2, an edge in an image is a contour of pixels of large gradient. Analogously, the average of gradient in such an image containing boundaries will have a positive value ξ . How big ξ is will determine the center of the distribution. It is difficult to pre-define this value. A dynamic approach should be considered. Let ξ be the gradient average of a reference image, the edge membership function with respect to G_{avg} can be represented as

$$\mu_{edge}(G_{avg}) = e^{-A_{e1}(G_{avg}-\xi)^2}, \quad (5.23)$$

where A_{e1} is a constant.

Regarding to G_{medi} and σ parameters, the edge membership functions are respectively

$$\mu_{edge}(G_{medi}) = e^{-A_{e2}(G_{medi}-\xi)^2}, \quad (5.24)$$

$$\mu_{edge}(\sigma) = e^{-A_{e3}(\sigma-\xi)^2}, \quad (5.25)$$

where A_{e2} and A_{e3} are constant, also determined by boundary conditions.

Case 3: Mixed-area Membership Function μ_{mixed}

For an image containing mixed area, there are no definite parameters such as G_{avg} , G_{medi} and σ that can be predicted. The parameters for such an image depend on the type and pattern of the mixed area. However, it can be easily ascertained by means of fuzzy set theory. According to the fuzzy set theory, at each point, the total degree should be one. For example, let the gradient average G_{avg} be a parameter, the following constraint should be satisfied:

$$\mu_{shade}(G_{avg}) + \mu_{edge}(G_{avg}) + \mu_{mixed}(G_{avg}) = 1. \quad (5.26)$$

Therefore, the mixed-area membership function can be derived from the shade and edge membership functions, that is,

$$\mu_{mixed}(G_{avg}) = 1 - \mu_{shade}(G_{avg}) - \mu_{edge}(G_{avg}), \quad (5.27)$$

$$\mu_{mixed}(G_{medi}) = 1 - \mu_{shade}(G_{medi}) - \mu_{edge}(G_{medi}), \quad (5.28)$$

$$\mu_{mixed}(\sigma) = 1 - \mu_{shade}(\sigma) - \mu_{edge}(\sigma). \quad (5.29)$$

The constants A_* can be determined by the boundary conditions. That is,

- (a) When $\mu_{shade}(0) = 1$, it is a pure shade-area, and then $\mu_{edge}(0) = 0$.
- (b) When $\mu_{edge}(\xi) = 1$, it is a boundary field, and then $\mu_{shade}(\xi) = 0$.

It may be noted that the Gaussian membership function is zero only when its parameter tends to be infinite. In practice, a threshold ϵ is assigned, where ϵ is a very small positive number. Consequently, by solving the boundary condition equations:

$$\begin{cases} \mu_{shade}(0) = 1 \\ \mu_{edge}(0) = \epsilon, \end{cases} \quad \begin{cases} \mu_{edge}(\xi) = 1 \\ \mu_{shade}(\xi) = \epsilon. \end{cases} \quad (5.30)$$

Then, the constants can be obtained by

$$A_{e1} = A_{s1} = \frac{\ln \epsilon}{\xi^2}. \quad (5.31)$$

So, the membership functions with parameter G_{avg} can be written as,

$$\mu_{shade}(G_{avg}) = e^{\frac{\ln \epsilon}{\xi^2} G_{avg}^2}, \quad (5.32)$$

$$\mu_{edge}(G_{avg}) = e^{\frac{\ln \epsilon}{\xi^2} (G_{avg} - \xi)^2}, \quad (5.33)$$

$$\mu_{mixed}(G_{avg}) = 1 - e^{\frac{\ln \epsilon}{\xi^2} G_{avg}^2} - e^{\frac{\ln \epsilon}{\xi^2} (G_{avg} - \xi)^2}. \quad (5.34)$$

Given $\epsilon = 0.01$ and $\xi = 128$, one sample of the membership function distribution of an image containing edge, shade or mixed area against G_{avg} are illustrated in Figure 5.4. From the figure, it can be clearly seen that an image having a

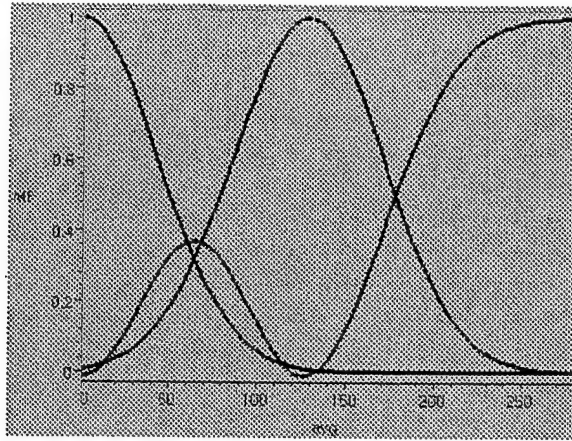


Figure 5.4: Membership functions with gradient average

particular value G_{avg} may be said to contain edge, shade or mixed area but with different degrees of membership.

3. Fuzzy Moment (FM) Descriptors

By taking the idea of fuzzy membership-distance from [4, 45], we define fuzzy moments between two images to evaluate the degree of the differences. A definition is given as follows.

Definition 5.3.1. Fuzzy shade moment $[FM_{I_1}^{I_2}]_{shade}$, is defined as the Euclidean distance of the membership values between two corresponding images I_1 and I_2 . That is,

$$[FM_{I_1}^{I_2}]_{shade} = \|\mu_{shade}^{I_1} - \mu_{shade}^{I_2}\|. \quad (5.35)$$

Simply, it can be rewritten as:

$$[FM]_{shade} = \|\mu_{shade}^{I_1} - \mu_{shade}^{I_2}\|. \quad (5.36)$$

Fuzzy edge and mixed moments can be obtained using this definition with only replacement of the term 'shade' by appropriate features.

4. Fuzzy Uncertainty Inference

Now, how does our intelligent approach perform image change detection? Since image understanding in human beings is also the result of reasoning in terms of knowledge about the world, uncertainty and fuzziness are inevitable problems. A fuzzy inferencing with uncertainty will be required.

In general, the fuzzy inference with uncertainty can be expressed as a IF-THEN fuzzy product rule:

$$\text{IF } \langle \text{Evidence} \rangle \text{ THEN } \langle \text{Strategy} \rangle (CF)$$

where $CF \in [0, 1]$ is a certainty factor indicating the certainty with which a fact is believed. A certainty factor of one indicates that it is very certain that a fact is true, and a certainty factor of zero indicates that it is very uncertain that a fact is true.

Uncertainty occurs when one is not absolutely certain about a piece of information. Also, uncertainty is referred as to the lack of adequate or correct information to make decision. Considered G_{avg} as a parameter, three fuzzy moments will be associated with change detection, i.e., $[FM(G_{avg})]_{shade}$, $[FM(G_{avg})]_{edge}$, $[FM(G_{avg})]_{mixed}$. Again, if all three parameters are considered, there are nine fuzzy moments. How do we make the final decision according to these moments? This reveals important deficiencies in areas such as the ability to detect inconsistencies in the knowledge. In order to make a reliable decision, a key factor CF will be used to evaluate the degree of certainty. Some key ideas relevant to the determination of CF are discussed as follows.

- **Detecting change based on shade feature**

This kind of detection will provide area change information. And the decision will particularly depend on the fuzzy shade moments, i.e., $[FM(G_{avg})]_{shade}$, $[FM(G_{medi})]_{shade}$, and $[FM(\sigma)]_{shade}$.

Notice that CF evaluates the certainty to believe that image change occurs and fuzzy moment indicates the degree of the difference. The larger the fuzzy moment, the more certain the change occurring. Based on these facts, it is acceptable to take the moment as a CF . For example, we can say:

The image area change has been detected based on the gradient average with certainty $CF_{shade}(G_{avg})$, where

$$CF_{shade}(G_{avg}) = [FM(G_{avg})]_{shade}. \quad (5.37)$$

- **Detecting change based on edge feature**

This kind of detection will associate with the boundary changes. The main factors includes the fuzzy edge moments such as $[FM(G_{avg})]_{edge}$, $[FM(G_{medi})]_{edge}$, and $[FM(\sigma)]_{edge}$.

The image boundary change has been detected based on the gradient average with certainty $CF_{edge}(G_{avg})$, where

$$CF_{edge}(G_{avg}) = [FM(G_{avg})]_{edge}. \quad (5.38)$$

- **Detecting change based on mixed-area feature**

From the definition of the mixed area, it has been realized that "mixed" is a very fuzzy concept. It is not easy to make a decision by only using mixed-area feature information. It should be combined with shade and/or edge feature information.

Based on the following consideration:

- (a) Combining the shade feature with mixed-area feature information (fuzzy moment), or the edge with mixed-area feature information provides a way for intelligent reasoning. In addition, the edge and shade features are considered as two independent features.
- (b) The mean and deviation are two important measures from the statistic point of view. One is for measuring the distribution of location, another is for the dispersion from mean. Single measures will lead to a bias in decision making. They should be considered simultaneously.

A hierarchical fuzzy inference model can be designed in the manner shown in Figure 5.5. Level 1 performs a fuzzy inference which is associated with statistic measures. An advanced fuzzy inference with respect to image features will be carried out at Level 2.

Table 5.1 shows the groups of the moments for determination of CF based on two statistic measures.

Table 5.1: Groups of the Fuzzy Moments for CF

CF	$[FM(\sigma)]_{shade}$	$[FM(\sigma)]_{edge}$	$[FM(\sigma)]_{mixed}$
$[FM(G_{avg})]_{shade}$	CF_{shade}^1		
$[FM(G_{medi})]_{shade}$	CF_{shade}^2		
$[FM(G_{avg})]_{edge}$		CF_{edge}^1	
$[FM(G_{medi})]_{edge}$		CF_{edge}^2	
$[FM(G_{avg})]_{mixed}$			CF_{mixed}^1
$[FM(G_{medi})]_{mixed}$			CF_{mixed}^2

It is easy to understand that the relationship between different groups is disjoint, and the relationship in the same group is conjunction. According to the fuzzy set theory, the conjunction and disjunction can be respectively defined as minimum and maximum of the involved facts. Therefore, the certainty factors can be determined by the following formulas:

$$\begin{aligned}
 CF_{shade}^1 &= \min\{[FM(\sigma)]_{shade}, [FM(G_{avg})]_{shade}\}, \\
 CF_{shade}^2 &= \min\{[FM(\sigma)]_{shade}, [FM(G_{medi})]_{shade}\},
 \end{aligned} \tag{5.39}$$

$$\begin{aligned}
 CF_{edge}^1 &= \min\{[FM(\sigma)]_{edge}, [FM(G_{avg})]_{edge}\}, \\
 CF_{edge}^2 &= \min\{[FM(\sigma)]_{edge}, [FM(G_{medi})]_{edge}\},
 \end{aligned} \tag{5.40}$$

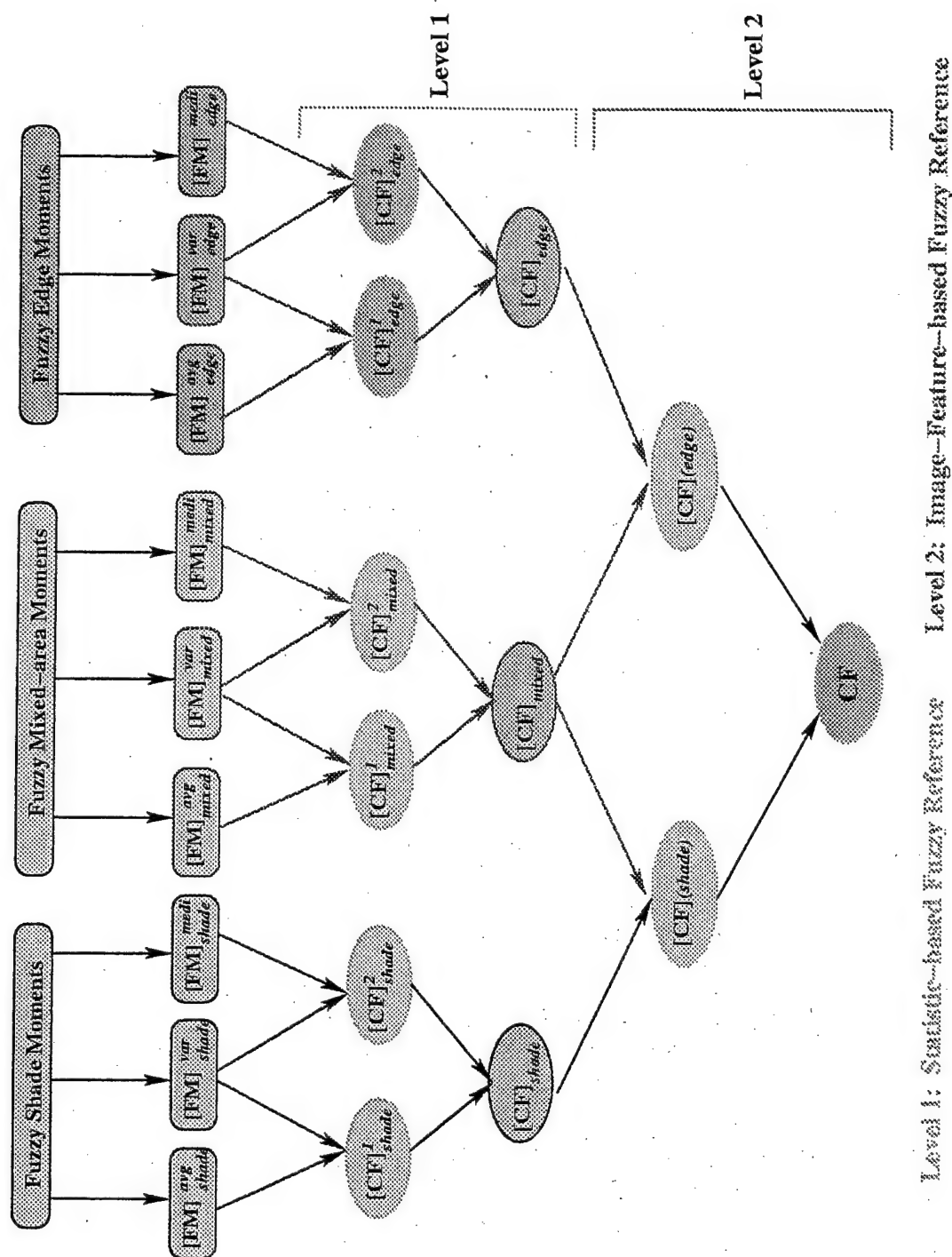


Figure 5.5: A hierarchical fuzzy inference model

$$\begin{aligned}
CF_{mixed}^1 &= \min\{[FM(\sigma)]_{mixed}, [FM(G_{avg})]_{mixed}\}, \\
CF_{mixed}^2 &= \min\{[FM(\sigma)]_{mixed}, [FM(G_{medi})]_{mixed}\}.
\end{aligned} \tag{5.41}$$

Therefore, the certainty factors for detecting feature changes can be given:

$$\begin{aligned}
CF_{shade} &= \max_i\{CF_{shade}^i\}, \\
CF_{edge} &= \max_i\{CF_{edge}^i\}, \\
CF_{mixed} &= \max_i\{CF_{mixed}^i\},
\end{aligned} \tag{5.42}$$

where $i = 1, 2$.

Up to now, a statistic-based fuzzy inference has been figured out. Since the mixed area is related to both shade and edge, an advanced fuzzy inference can be carried out by integrating the mixed-area information in shade or edge change detection. More details are given as follows.

A certainty factor for shade change detection can be calculated

$$CF(shade) = \alpha CF_{shade} + (1 - \alpha) CF_{mixed}. \tag{5.43}$$

In the same way, a certainty factor for boundary change is

$$CF(edge) = \beta CF_{edge} + (1 - \beta) CF_{mixed}, \tag{5.44}$$

where α and β are weights, in practice, they are determined by experience.

Finally, the certainty factor with which the image changes have been detected can be obtained by

$$CF = \max\{CF(shade), CF(edge)\}. \tag{5.45}$$

The certainty factor CF indicates how many degrees image changes can be trusted. Consequently, the larger the certainty factor is, the higher the degrees is.

Chapter 6

EVALUATION AND RESULT ANALYSIS

Detecting and representing changes to data is important for active GIS databases. The previous chapter focuses on developing the change detection algorithm by utilizing satellite images. In this chapter, the theoretical analysis will be provided. The further evaluation is investigated by using real images. Finally, the hybrid change detection approach is summarized.

6.1 Theoretical Analysis of Image Change Detection

For theoretical evaluation, a simple image with regular object shapes will be considered.

Histogram Analysis

Given an image with five objects that have regular shape shown in Figure 6.1(a1). Different colors are just used to demonstrate different gray-scale.

Histograms can be calculated straightforwardly. A simple process can be summarized as follows:

- set the whole list to zero;
- scan all pixels of the image, and take the gray value as the index to the list;
- increment the corresponding element of the list by one.

The corresponding raster model and plot of histogram are shown in 6.1(a2) and 6.1(a3), respectively. Elementary classification of objects within the image scene is clearly illustrated in Figure 6.1(a3). Structure change can be carried out by comparing Figure 6.1(a) and 6.1(b). Moreover, the deficiency to detect two objects with the same gray values is also illustrated in Figure 6.1(a) and 6.1(c).

This simple analysis shows that the histogram analysis fails to detect the change when objects have the same brightness values. Actually, the statistical significance of a result is the probability that a difference occurred. In order to provide additional information, an advanced change detection should be performed.

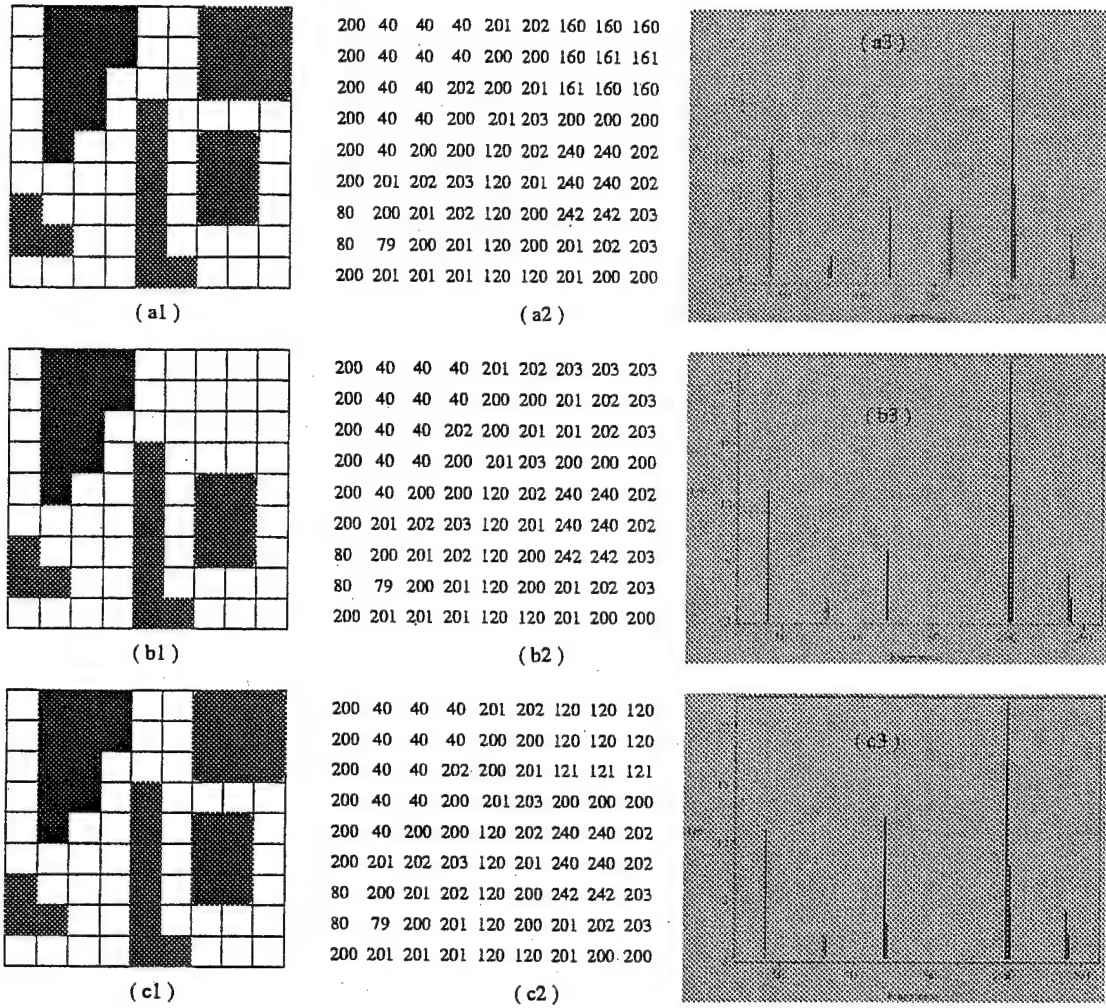


Figure 6.1: An example of histogram analysis

Fuzzy Intelligent Analysis

In order to evaluate the fuzzy intelligent detection algorithm, the same example shown in Figure 6.1 will be further used. After the feature extraction, the basic measures are provided as:

1. For the image I_1 in Figure 6.1 (a1), $G_{avg}(I_1) = 44.187$, $\sigma(I_1) = 33.875$, and $G_{medi}(I_1) = 56.23$;
2. For the image I_2 in Figure 6.1 (b1), $G_{avg}(I_2) = 41.270$, $\sigma(I_2) = 36.045$, and $G_{medi}(I_2) = 56.23$;

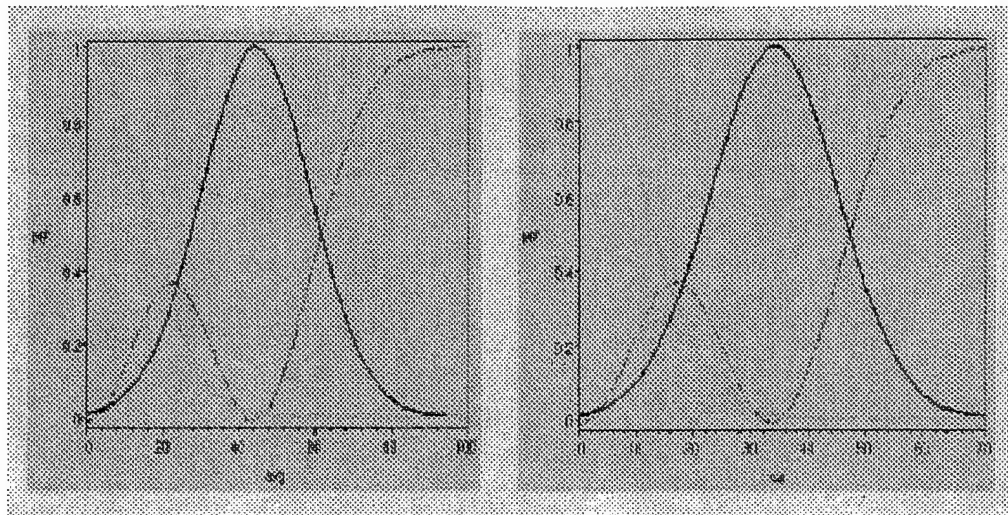


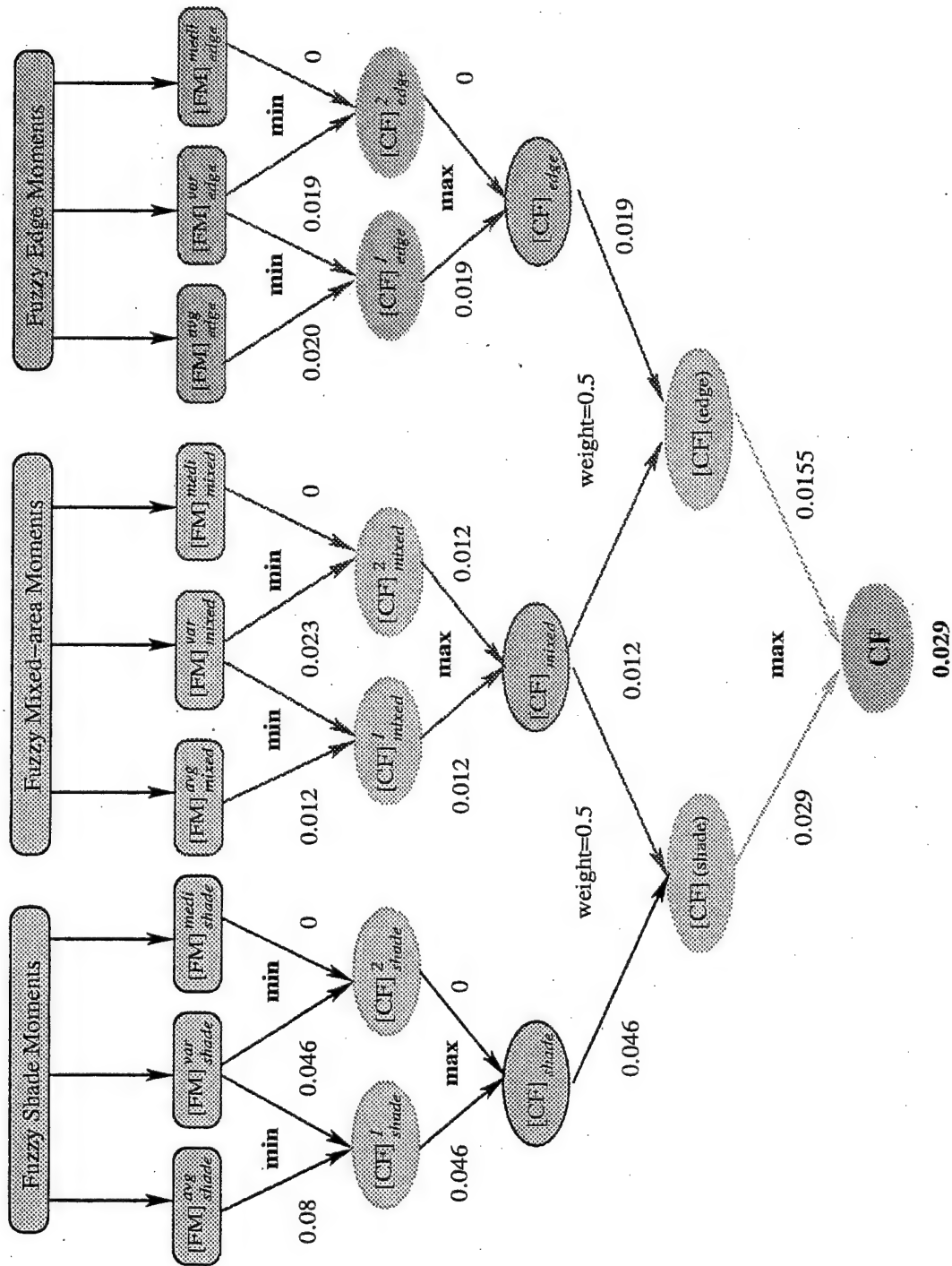
Figure 6.2: Membership functions for theoretical analysis

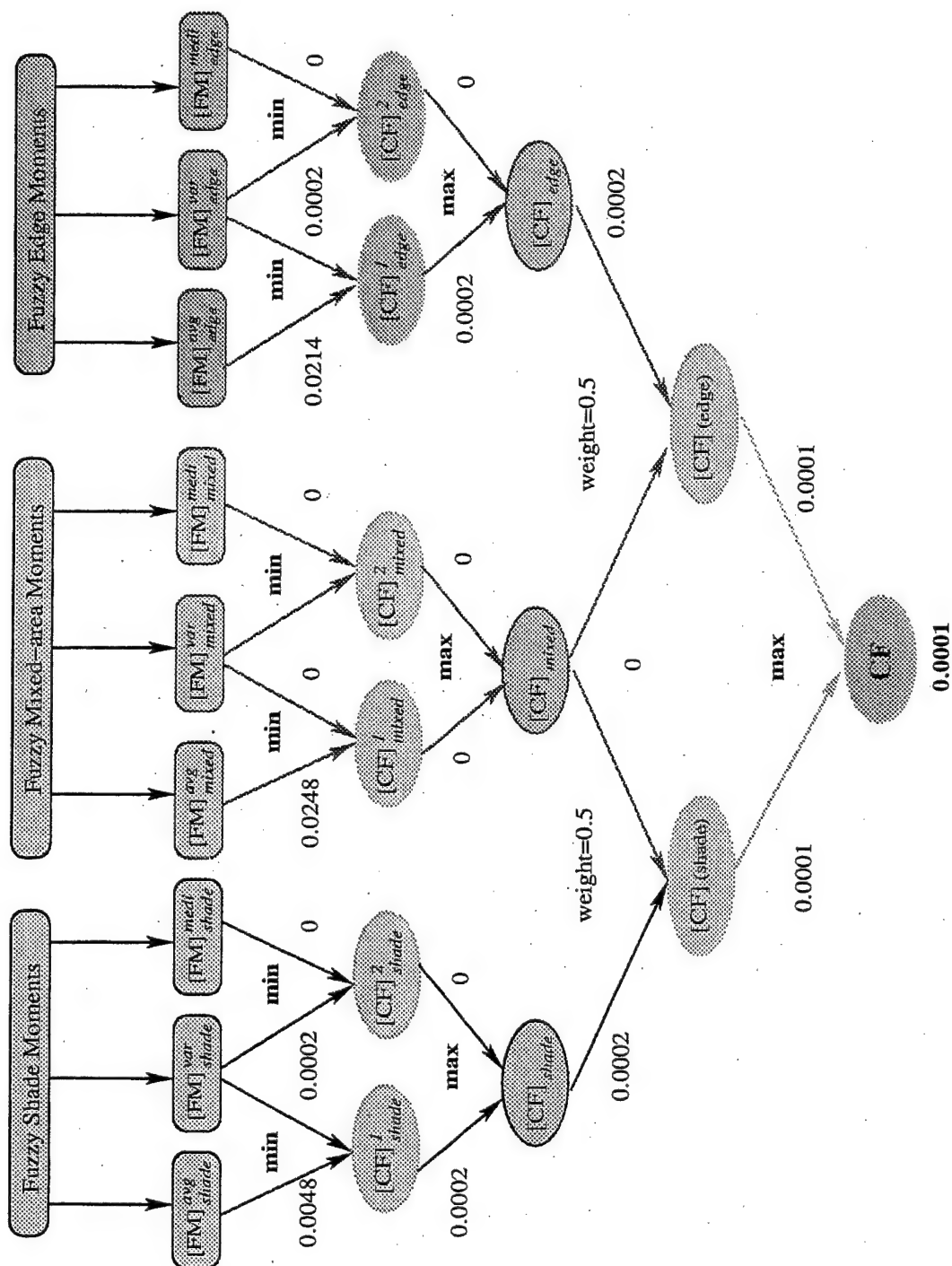
3. For the image I_3 in Figure 6.1 (c1), $G_{avg}(I_3) = 47.216$, $\sigma(I_3) = 33.129$, and $G_{medi}(I_3) = 56.23$.

Let the image I_1 be a reference image or base image. Given $\epsilon = 0.01$. Then the membership functions that describe an image contains edge, shade or mixed area against mean G_{avg} and standard deviation σ are illustrated as Figure 6.2. The fuzzy uncertainty inference structure is shown as in Figure 6.3 by comparing the image I_1 and image I_2 . Figure 6.4 shows the fuzzy uncertainty inference result by comparing the image I_1 and image I_3 .

It is noticed that there exist a big difference between the resulting certainty factors 0.029 and 0.0001 quantitatively, which indicate that the intelligent algorithm based on the fuzzy inference can perform a better change detection than the statistical analysis method. Specifically, it can detect two objects with the same gray-scale values.

Although the analysis is based on a simple image consisting of just 9×9 pixels, this example provides an insight into the utility of this intelligent image change detection. It makes sense that fuzzy set theory is not a panacea[2], but it does offer a significant potential for image change detection.

Figure 6.3: A hierarchical fuzzy uncertainty inference for image I_1 and image I_2

Figure 6.4: A hierarchical fuzzy Uncertainty inference for image I_1 and image I_3

6.2 Evaluation via Real Image Data

The real raster data used for evaluation of the detection algorithm has been selected from 2D/3D City Digital map database [60]. For the purpose of the structure change detection, only histogram analysis is provided for the pre-detection.

Histogram Analysis

Our special consideration is also the gray level histogram for pre-detection, which gives us a graphical representation of how many pixels within an image fall into the various gray level boundaries.

The Figure 6.5 gives a picture for the pre-detection, where,

Figure 6.5 (a1) is Ground Topography Raster Data.

Figure 6.5 (b1) is Ground Topography and Building Height Raster Data.

Figure 6.5 (c1) is Morphological Clutter Raster Data which includes building, open area, forest, and water

And Figure 6.5 (a2), (b2) and (c2) are the grey-scale images, respectively. Figure 6.5 (a3), (b3) and (c3) are corresponding histograms of the grey-scale images.

Based on the histogram of an image, it can be seen immediately whether the image is basically dark or light and high or low contrast.

The peaks in the histogram shown in Figure 6.5(c3) are associated with "building, forest, water and open area". The histogram analysis shows that there are at least four different types of objects within that image.

Comparing the histograms of different images which cover the same area gives the pre-detection result, that is changes have occurred statistically. However, the judgment of trustiness on change requires to perform the further analysis.

Fuzzy Intelligent Analysis

Figure 6.6 shows the original images and corresponding feature images which are derived by the edge extraction.

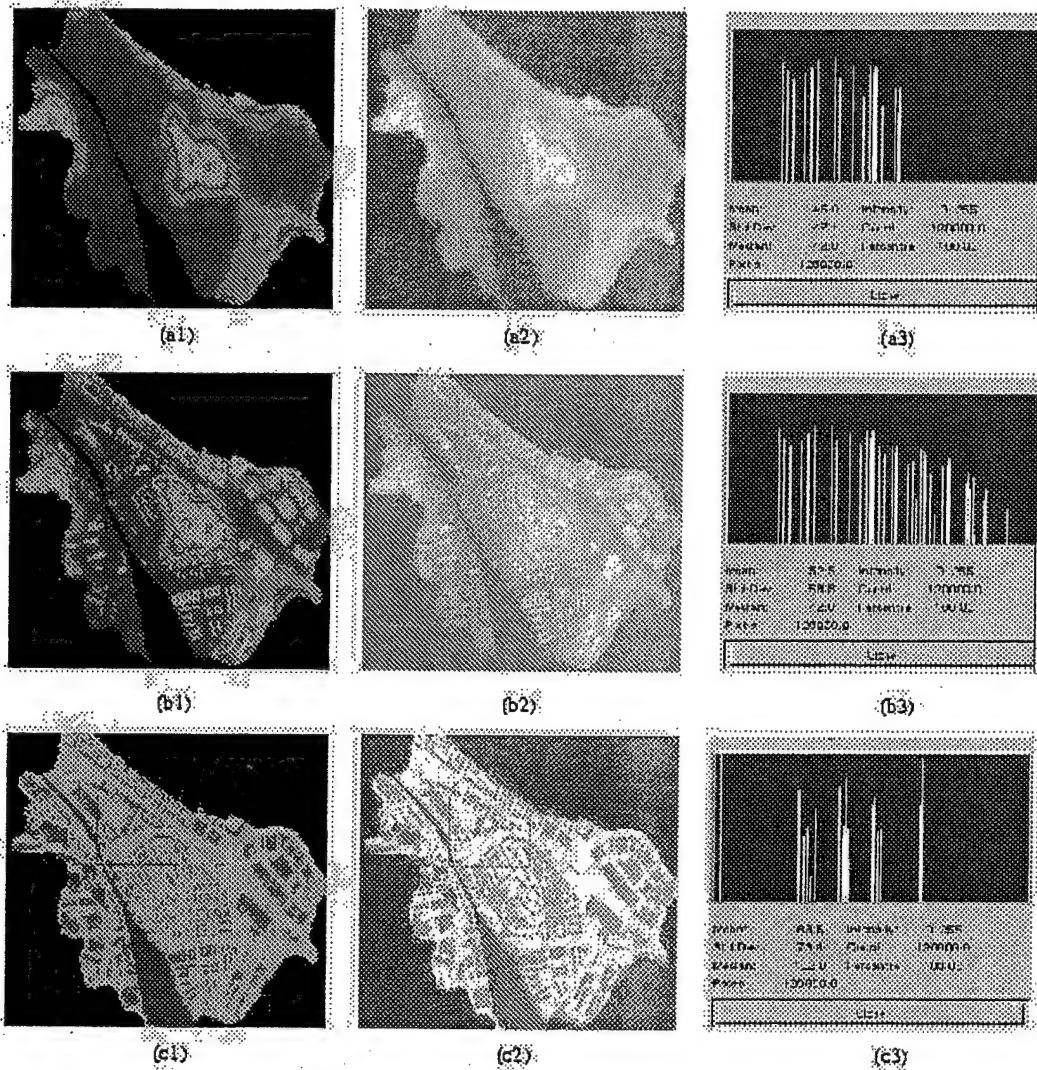


Figure 6.5: Pre-detection for real image data

After the extraction of image features is performed, the following statistical data for fuzzy intelligent detection can be calculated.

1. In case shown in Figure 6.6(a), the gradient average $G_{avg} = 7.9$, standard deviation $\sigma = 19.9$, and median value $G_{medi} = 0$.
2. In case shown in Figure 6.6(b), the gradient average $G_{avg} = 15.4$, standard deviation $\sigma = 31.7$, and median value $G_{medi} = 0$.

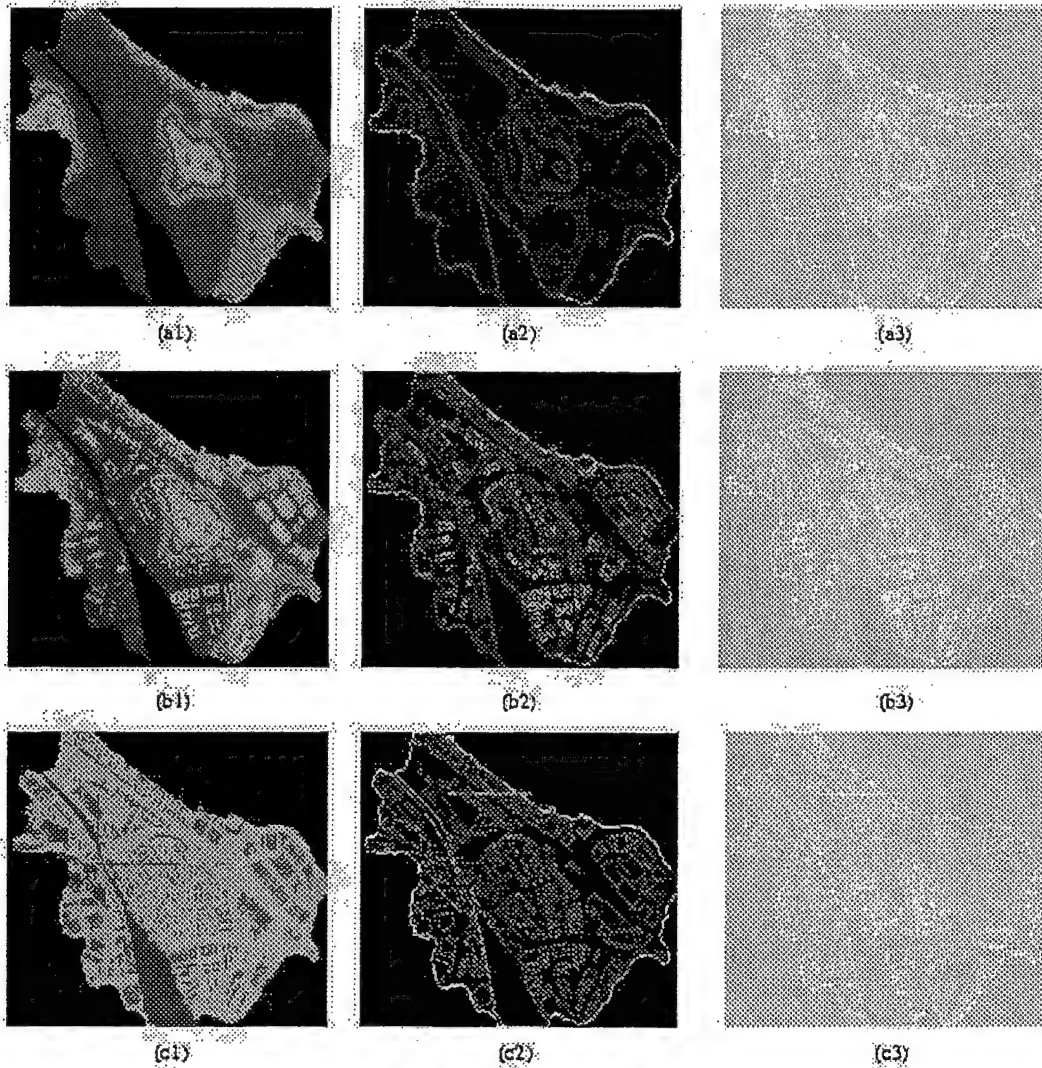


Figure 6.6: Real image feature extraction for post-detection

3. And for the Figure 6.6(c), the gradient average $G_{avg} = 18.7$, standard deviation $\sigma = 39.6$, and median value $G_{medi} = 0$.

Based on these parameters, the intelligent change detection can be analyzed. Let the image shown in Figure 6.6(b) be a reference image. Then the membership functions with respect to the gradient average and standard deviation are given in Figure 6.7 and Figure 6.8, respectively. By comparing the images in Figure 6.6 and (b), the fuzzy inference is represented as in the Figure 6.9.

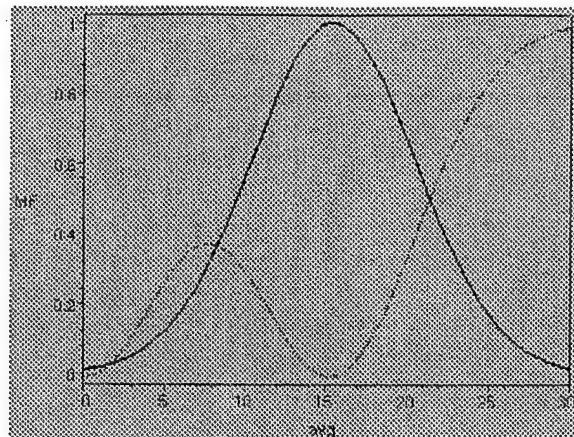


Figure 6.7: Membership functions with gradient average

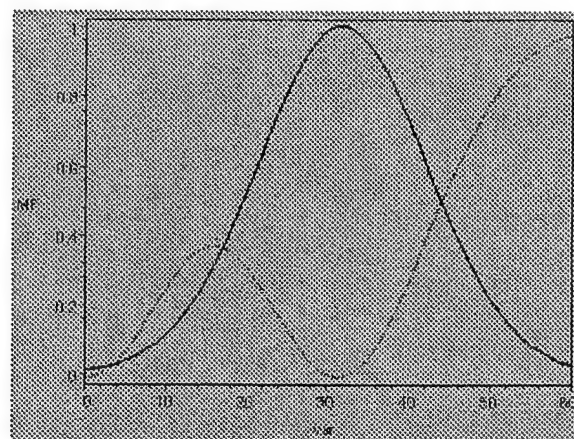


Figure 6.8: Membership functions with standard deviation

6.3 Real Implementation Consideration

Successful image change detection based on satellite data requires many facts. Ideally, the image data used to perform change detection should be acquired by a remote sensor system that holds the basic resolution constants such as temporal, spatial, spectral and radiometric.

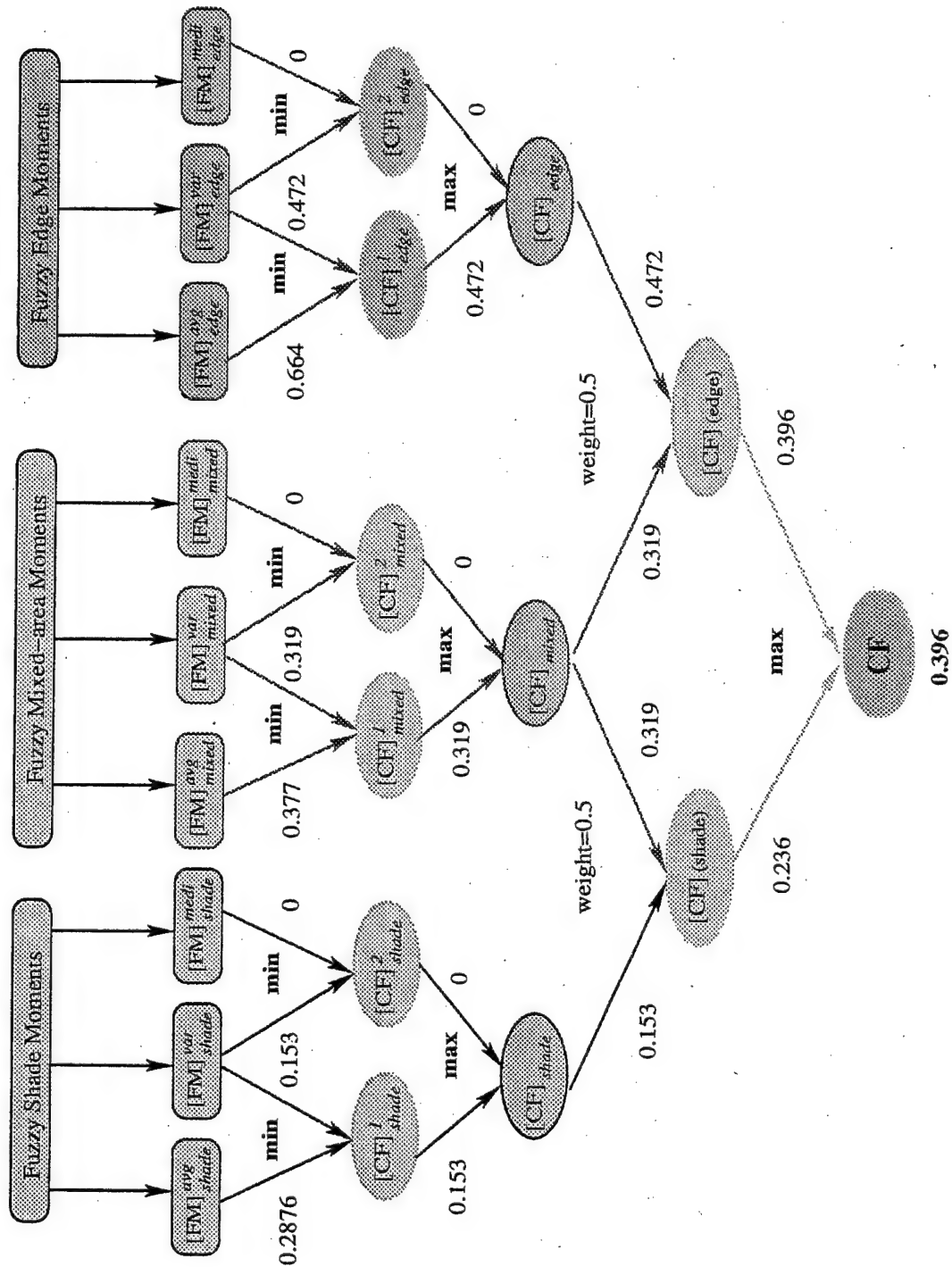


Figure 6.9: A hierarchical fuzzy inference for real image

Temporal Resolution: the data used to perform image change detection should be acquired at approximately the same time of day. This eliminates diurnal sun angle effects. Whenever possible, it is desirable to use data acquired on anniversary dates. In this way, seasonal sun angel and plant phonological difference that can destroy the change detection can be removes.

Spatial Resolution: Although it is possible to perform change detection using data collected from two different sensor systems with different instantaneous field of views (IFOV), ideally, the data are acquired by a sensor system that collects data with the same IFOV.

Also environmental characteristics are important when performing change detection. For example, atmospheric and soil moisture conditions, and phonological cycle will affect the change detection. It is desirable to hold environmental variables as constant as possible.

Although the intelligent change detection has been not completely implemented in a real system, the theoretical analysis and real raster data evaluation show that it can achieve reasonable successes to evaluate the change observed in the scene. However, it is still necessary to do further investigation by applying it to a real-world problem. This is because this evaluation is only loosely associated with real system, and also real implementation requires additional transformation of data between different formats such as vector to raster.

6.4 Summary of the Hybrid Approach

The hybrid method for image change detection can be summarized as shown in Figure 6.10.

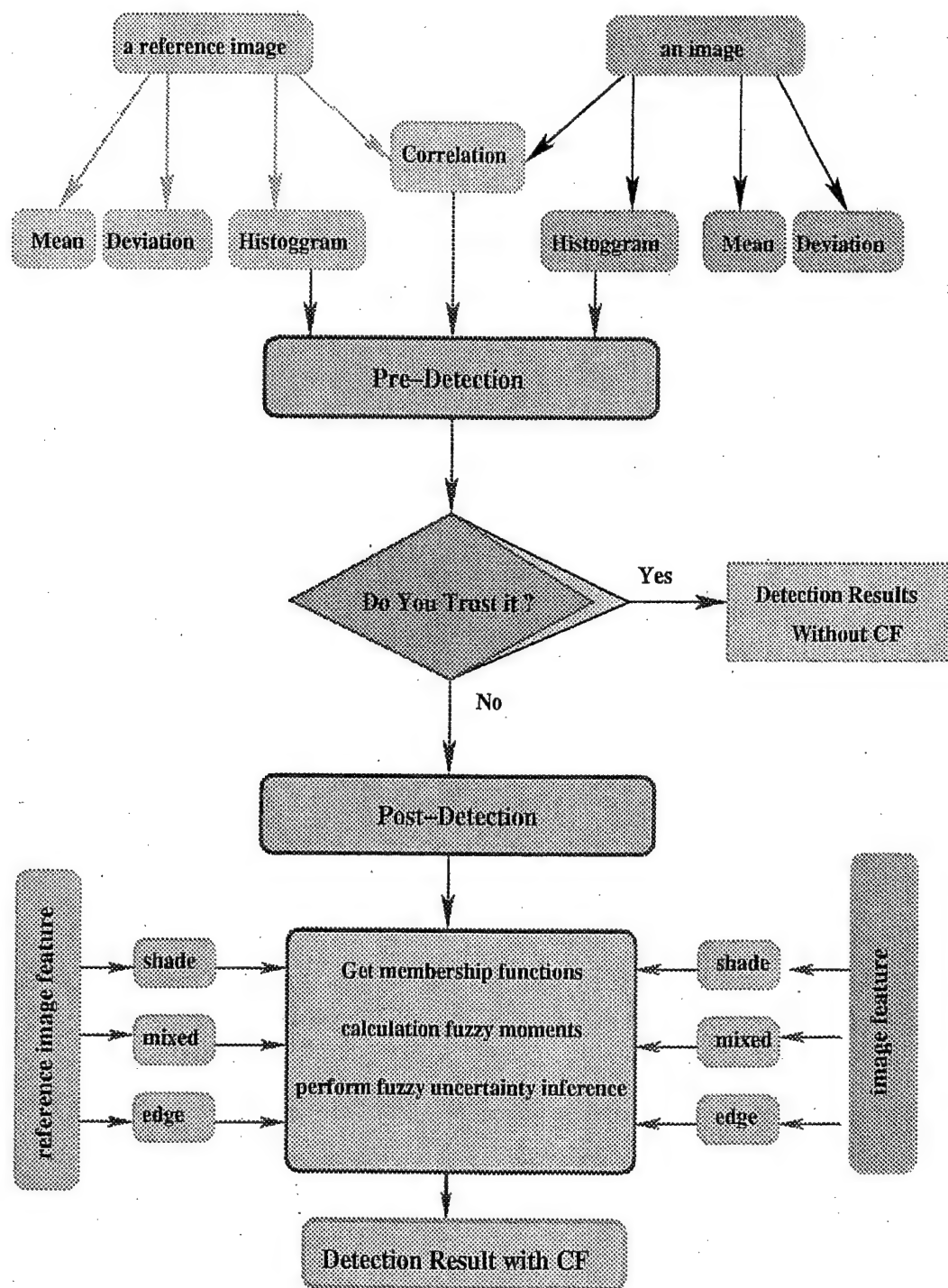


Figure 6.10: Summary of the hybrid image change detection

Chapter 7

CONCLUSION

In a distributed environment, compatibilities and consistencies are major concerned issues. The technical approach to handling these issues is referred as to "conflation". Right now, the conflation is becoming such a broader consideration in GIS that most GIS-oriented organizations need conflation technology, even if many of them do not yet fully understand its role. The payoff of successful conflation in better meeting GIS data needs is enormous. Therefore, to investigate or to solve conflation problems in the distributed intelligent mobile agent system is of great significance.

Since change over time in geographic area is particularly important in GIS application, in the recent years there has been a significant amount of research put forth in the development of change detection methods. However, a more direct approach is to seek information resources that record the changes directly. The image data from the satellite have been proved to be extremely useful for change detection techniques. It is not difficult to see that, indeed, image change detection has a great potential in augmenting the regular conflation.

In the conclusion, the main contributions or results are emphasized. The open research issues are discussed, and the future work is provided.

7.1 Main Contributions

With respect to conflation and image change detection issues in GIS, the main contributions of the dissertation include:

- **A flexible conflation model**

In order to remedy a defect of the regular conflation algorithms, the major gaps or shortfalls in the conflation are identified, and then a raster-based vector conflation model is developed. The main idea is to use image change detection to enhance the conflation capabilities. It is apparent that the model differs significantly from the previous attempts, inasmuch as the detecting process is using the real time

satellite image data. In this way, a vector-based GIS system may augment the conflation with image change detection in the raster domain by performing a vector-to-raster conversion. Also, the model is more general since it can work in two different ways, that is, pre-detection/post-conflation and pre-conflation/post-detection.

- **A conflation scheme based on vector data**

Based on the conflation model developed for distributed mobile agent systems, the dissertation is specially focused on processing spatial and non-spatial conflation problems in vector GIS. By emphasizing that conflation is an inexact process, a component-ware conflation scheme is investigated, in which spatial conflation is performed by mainly using the mathematical tool such as distance measures, and non-spatial conflation is performed by employing the artificial intelligent techniques. It is possible to assemble these components to process any specific problems.

- **A hybrid change detection approach**

Since image interpretation frequently involves human knowledge which inherits the incompleteness and imprecision, the problems associated with the areas of fuzziness and uncertainty are of great concern in image change detection. An important contribution of the work is to develop a hybrid approach for image change detection by taking the advantages of statistical analysis methods and fuzzy inference. That is a two-step method, in which the intelligent change detection can perform a fuzzy inference under uncertainty.

As the complexity and intelligence of GIS application are progressing, the utility of intelligent techniques such as fuzzy sets will increase. Incorporating fuzzy sets in GIS is becoming cost-effective approach to address complex problems. The novelty of the approach presented here is to employ more potential techniques – the fuzzy uncertainty inference. A hierarchical inferencing structure presented indicates a simple way to perform a fuzzy inference. The approach is examined by its ability to support a simple image change detection. Furthermore, the test on real images have shown that it is capable of detecting meaningful changes.

7.2 Open Research Issues

The conflation is a practical and complex process. Based on the model presented, the focus of the dissertation is put on the spatial and non-spatial conflation in vector data, and specially on the image change detection for augmenting the conflation. The real implementation will also require many other steps such as transformation between different format, vectorization of the raster data, and development of knowledge bases, etc. All of these are open research areas. They require sufficient investigations.

7.3 Future Work

Even though the dissertation developed a flexible conflation model, the implementation remains a challenging area for further investigation since the real world is very complicated. The following discussions will provide some research ideas for improving the change detection in the future.

- **Modification of the membership functions**

The specification and tuning of membership functions has been a source of much criticism leveled at the fuzzy logic approach [41]. Often the method of specifying the membership function is summarily described as being chosen by experts [1]. There are amount of researches and applications that involves specifying the membership functions in GIS, such as, using fuzzy membership functions from experiment with GISs user for decision support [33], extracting fuzzy membership functions from the volumes of historical data for a contextual fuzzy cognitive map framework [52].

It is important to note that the membership functions in the intelligent image change detection are defined by interpreting the images, which also depend on the experience. The further modification of membership functions should be able to improve the fuzzy intelligent detection.

- **Determination of the weights**

For performing the reasoning under the uncertainty in the fuzzy intelligent detection, the weights at level two have to be pre-defined. However, the determination of the weights in the hierarchical inferencing structure is still free. How to design a

knowledge base for weights is a potential way to improve the innovative algorithm presented here.

BIBLIOGRAPHY

- [1] Genst A. De. F. Canters, and H. Gulink. Uncertainty modeling in buffer operations applied to connectivity analysis. *Transactions in GIS*, 5(4):305-326, 2001.
- [2] ACM. Coping with the imprecision of the real world: An interview with lotfi a. zadeh. *Communications of the Association of Computing Machinery*, 27:304-311, 1984.
- [3] Nabil R. Adam and Aryya Gangopadhyay. *Database Issues In Geographic Information Systems*. Kluwer Academic, 1997.
- [4] Bijita Biswas, Amit Konar, and Achintya K. Mukherjee. Image matching with fuzzy moment descriptors. *Engineering Applications of Artificial Intelligence*, 14(1):43-49, 2001.
- [5] Peter A. Burrough and Andrew U. Frank. *Geographic Objects with indeterminate Boundaries*. Taylor & Francis Ltd, 1996.
- [6] Nicholas Chrisman. *Exploring Geographic Information Systems*. John Wiley & Sons, Inc., 1997. Printed in the United States of America.
- [7] M. Cobb, Miyi J. Chung, and *et al.* A rule-based approach for the conflation of attributed vector data. *GeoInformatica*, 2(1):7-35, 1998.
- [8] M. Cobb, F. Petry, and K. Shaw. Uncertainty issues of conflation in a distributed environment. Paper, GIS/LIS, November 1998.
- [9] Maria A. Cobb, Dia L. Ali, and *et al.* Intelligent database agents for geospatial knowledge integration and management. Project, National Imagery and Mapping Agency (NIMA), University of Southern Mississippi, <http://sdiagent.st.usm.edu>, 2001.
- [10] Peter J. Deer. Digital change detection techniques: Civilian and military applications. Report, NASA, <http://ltpwww.gsfc.nasa.gov/ISSSR-95/digitalc.htm>, 2001.
- [11] Delaware. The impact of census block conflation on census 2000 related projects in delaware county. Paper, GISCAFE, <http://www.giscafe.com/TechPapers/Papers/paper044>, 2001.
- [12] S. Dellepiane and G. Vernazza. Model generation and model matching of real images by a fuzzy approach. *Pattern Recognition*, 25(2):115-137, 1992.
- [13] Edward R. Dougherty and Phillip A. Laplante. *Introduction to Real-time Imaging*. SPIE Optical Engineering Press and IEEE Press, Bellingham, Washington USA, 1995.
- [14] P. Dre and J. Ying. Geochange: An experiment in wide-area database services for geographic information exchange. In *Proceedings, Third Forum of Research and Technology Advances in Digital Libraries*, pages 14-23, 1996.
- [15] ESRI. Arc/info users guide, release 7. Technical report, Environmental Systems Research Institute, Redlands, CA, 1994.

- [16] C. Fischer, J. Frew Larsgaard, and *et al.* Alexandria digital library: Rapid prototype and metadata schema in advances in spatial databases. In *4th International Symposium*, pages 184-195, 1995.
- [17] Li Min Fu. *Neural Networks in Computer Intelligence*. McGraw-Hill, Inc., 1994.
- [18] Donald Geman and *et al.* *Spatial Statistics and Digital Image Analysis*. National Academy Press, Washington, D.C, 1991.
- [19] Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing*. Addison-Wesley Publishing Company, 1977.
- [20] Michael F. Goodchild. Uncertainty: The achilles heel of gis. *Geographic Information Systems*, Nov. 1998.
- [21] K. Greene, B. Kempka, and L. Lackey. Using remote sensing to detect and monitor land-cover and land -use change. *Photogrammetric Engineering and Remote Sensing*, 60(3):331-337, 1994.
- [22] Kathleen Hornsby and Max J. Egenhofer. Identity-based change: a foundation for spatio-temporal knowledge representation. *Int. J. Geographical Information Science*, 14(3):207-224, 2000.
- [23] Jang J.-S. R. Sun C.-T. and E. Minzutani. *Neuro-Fuzzy and soft computing: A computational approach to learning and machine Intelligence*. Prentice Hall, Upper Saddle River, New Jersey, 1998.
- [24] Bernd Jahne. *Practical Handbook on Image Processing for Scientific Applications*. CRC Prss, Boca Roton, New York, 1997.
- [25] Jensen John R. *Introduction Digital Image Processing: a remote sensing perspective*. Prentice-Hall, Inc., Simon and Schuster/A Viacom Company, Upper Saddle River, NJ07458, 1996.
- [26] Jensen J.R. and Schill S.R. 'General steps required to perform change detection. Technical report, GISDEVELOPMENT, <http://www.gisdevelopment.net/technology/datacon/techdc0004.htm>, 2001.
- [27] Kwang In Kim, Keechul Jung, and Hang Joon Kim. Face recognition using kernal principal component analysis. *IEEE Signal Processing Letters*, 9(2), 2002.
- [28] George B. Korte. *The GIS Book*. Word Press, 2001.
- [29] Zadeh L.A. Fuzzy sets. *Information and Control*, pages 338-353, 1965.
- [30] Zadeh L.A. Outline of a new approach to analysis of complex systems and decision processes. *IEEE Transactions on system, Man and Cybernetics*, 3(1):28-44, 1973.
- [31] Bang W. Lee, Jin W. Lee, Si H. Bae, and Suh K. Kim. Programmable facsimile image processor including fuzzy-based decision. In *IEEE*, pages 482-485, 1993.
- [32] Jay Lee and Wong David W.S. *Statistical Analysis with Arcview GIS*. John Wiley & Sons, INC., 605 Third Avenue, New York, NY 10158-0012, 2001.

- [33] Zhi-Qiang Liu and Richard Satur. Contextual fuzzy cognitive map for decision support in geographic information systems. *IEEE Transactions on Fuzzy Systems*, 7(5):481-494, 1999.
- [34] Users Manual. Change detection tutorial. Technical report, AISRI, <http://www.ai.sri.com/~radius/testbed/user-manual/node56.html>, 2001.
- [35] J. McNeely. Automated conflation and update of geospatial feature data bases. In *ESRI User Conference*, 1997.
- [36] Koji Miyajima and Anca Ralescu. Modeling of natural objects including fuzziness and application to image understanding. In *IEEE*, pages 445-450, 1993.
- [37] Carlotto M.J. Detection and analysis of change in remotely sensed imagery with application to wide area surveillance. *IEEE Transactions on Image Processing*, 6(1), 1997.
- [38] Ralph O. Mueller. *Basic Principles of Structural Equation Modeling*. Springer-Verlag, New York, 1996.
- [39] Wayne Niblack. *An Introduction to Digital Image Processing*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1986.
- [40] Department of Defense. Interface standard for vector product format. Technical report, MIL-STD-2407 and MIL-STD-600006, 1996.
- [41] Burrough P.A. Van Ganns P.F.M. and MacMillan R.A. High-resolution landform classification using fuzzy k-means. *International Journal of Fuzzy Sets and Systems*, 113(1):37-52, 2000.
- [42] Kyriakidis P.C. Shortridge A.M. and Goodchild M.F. Geostatistics for conflation and accuracy assessment of digital elevation models. *Int. J. Geographical Information Science*, 13(7):677-707, 1999.
- [43] Fisher P.F. and S. Pathirana. The evaluation of fuzzy membership of land cover classes in the suburban zone. *Remote Sensing of Environment*, 34:121-132, 1990.
- [44] Team Project. Satellite image processing. Paper, Stanford, <http://www-ise.Stanford.edu/class/ee368/project21/ee368writeup.htm>, 2001.
- [45] B. Ramamurthy and A. Gershe. Classified vector quantization of images. *IEEE Transaction on Communications*, Com-34(11):1105-1115, 1986.
- [46] Gonzalez R.C. and P. Wintz. *Digital Image Processing*. Addison-Wesley, Reading, MA, 1993.
- [47] Ali Rejaie and Masanobu Shinozuka. Damage assessment from remotely sensed images using pca. In *SPIE's 8th Annual International Symposium on Smart Structures and Materials*, Newport Beach, CA., 2002. March 4-8.
- [48] Team Report. Wide area surveillance-innovative techniques for change detection. Technical report, Veridian, <http://www.veridian.com/services/wascd.asp>, 2001.

- [49] A. Rosenfeld. *Topics in Applied Physics: Digital Picture Analysis*. Springer-Verlag Berlin Heidelberg, New York, 1976.
- [50] Neil C. Rowe and Lynne L. Grewe. Change detection for linear features in aerial photographs using edge-finding. Paper, NAVY, <http://www.cs.nps.navy.mil/people/faculty/rowe/diffpap.html>, 2001.
- [51] A. Saalfeld. Conflation: Automated map compilation. *International Journal of GIS*, 2(3):217-228, 1988.
- [52] Richard Satur and Zhi-Qiang Liu. A contextual fuzzy cognitive map framework for geographic information systems. *IEEE Transactions on Fuzzy Systems*, 7(5):495, 1999.
- [53] B. Scholkopf, A. Smola, and K. Muller. Non-linear component analysis as a kernel eigenvalue problem. *Neural Computing*, 10:1299-1319, 1998.
- [54] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
- [55] A. Singh. Review article: Digital change detection techniques using remotely sensed data. *Int. Journal of Remote Sensing*, 10(6):989-1003, 1989.
- [56] Paul Sutton. The ncgia gis core curriculum for technical programs. Unit11: Registration and conflation, University of California, <http://www.ncgia.ucsb.edu/cctp/units/unit11/11.html>, 2001.
- [57] Altlas Team. Atlas geomatics: Data integration. Project, ATLAS, http://www.atlas.co.il/data_integration.htm, 2001.
- [58] USGS Team. Usgs science for changing world. Project, USGS, <http://edcnts12.cr.usgs.gov/ned-h/about/conflation.html>, 2001.
- [59] F. Wang. Improving remote sensing image analysis through fuzzy information representation. *Photogrammetric Engineering and Remote Sensing*, 56(8):1163-1169, 1990.
- [60] Website. 2d/3d city digital map database. Technical report, DISCMAP, <http://www.discmap.se/city.htm>, 2002.
- [61] R. Wiemker. An iterative spectral-spatial bayesian labeling approach for unsupervised robust change detection on remotely sensed multispectral imagery. In *Proceedings of the 7th International Conference on Computer Analysis of Images and Patterns*, pages 263-270. Springer LNCS 1296, 1997.
- [62] Pratt W.K. *Digital Image Processing*. Wiley-Interscience, New York, 1978.
- [63] H. Wu, Q. Chen, and M. Yachida. Face detection from color images using a fuzzy pattern matching method. *IEEE Transaction on Pattern Matching and Machine Intelligence*, 21(6):557-563, 1999.
- [64] Shuxin Yuan and Chuang Tao. Development of conflation components. In *Proceedings of Geoinformatics'99 Conference*, pages 1-13, Ann Arbor, 1999. Geo-informatics and Socio-informatics.

INDEX

- AI, 1
- AM, 17
- Atlas, 5
- attribute transfer, 8
- CA, 17
- change detection, 10, 12
- compatibility, 63
- component-ware technology, 23
- conclusion, 63
- conflation, 2, 4, 5, 16, 17, 22, 63
- conflation classification, 7
- conflation method, 9
- conflation model, 18, 63
- consistency, 63
- correlation, 32, 38, 40
- correlation analysis, 12
- cross-classification, 11
- DALIS, 5
- data conflict, 7
- data converseion, 15
- data overlay, 7
- deconfliction, 10
- Dempster-Shater theory, 9
- distributed mobile agent model, 16
- DMAP, 6
- edge, 31
- edge matching, 7
- edge membership function, 43
- EDNA, 5
- Euclidean distance, 27
- evidential reasoning uncertainty, 9
- feature matching, 9
- Fourier descriptor, 27
- fuzzy moment, 45
- fuzzy set, 34, 35
- fuzzy uncertainty inference, 46
- GIS, 1, 6, 51, 63
- Global Position System, 9
- gray scale, 39
- Hausdorff distance, 27
- histogram, 38, 40
- image change detection, 10, 30, 37, 63
- image deviation, 11
- image differencing, 11
- image-image conflation, 8
- incompatibility, 2, 4
- inconsistency, 2, 5
- Local Cadastral System, 9
- mean value, 32
- membership function, 35, 65
- MF, 35
- mixed-area membership function, 44
- mixed-range, 31
- mobile agent, 1, 2, 16
- normalization, 39
- NRL, 6
- positional re-alignment, 9
- post-conflation, 21
- post-detection, 21
- pre-conflation, 21
- pre-detection, 21
- Principal Components Analysis, 12
- probability density function, 38
- QA, 17
- QM, 16
- RA, 17
- raster data, 14, 31
- result analysis, 51
- ROI, 16
- shade, 31
- shade membership function, 43

spatial discrepancy, 8
spatial feature transfer, 8
spatial reference system, 7
standard deviation, 32
statistic analysis, 37
statistic measures, 32

TIGER, 5

uncertainty, 6, 20
Universal Transverse Mercator, 9
universe of discourse, 35
USGS, 5

variance, 32
vector conflation, 6
vector data, 14
vector-image conflation, 8
vector-vector conflation, 8

ISSN 0165-0114

FUZZY

sets and systems

International Journal of Soft Computing and Intelligence

official publication of the
International Fuzzy Systems Association

IFSA

Volume 113 Number 1 July 1, 2000

Special Issue

Uncertainty in Geographic Information Systems
and Spatial Data

Your First-Stop Source for
Computational
Intelligence Information
and Services on the Web
www.elsevier.com/cite

COMPUTATIONAL
INTELLIGENCE
CITE

north-holland

FUZZY SETS AND SYSTEMS

SPECIAL ISSUE

Uncertainty in Geographic Information Systems
and Spatial Data

Guest Editors

Maria Cobb, Fred Petry and Vince Robinson

FUZZY SETS AND SYSTEMS
VOLUME 113, NUMBER 1, JULY 1, 2000

Special Issue: Uncertainty in Geographic Information Systems and Spatial Data

CONTENTS

<i>M. Cobb, F. Petry and V. Robinson</i> , Preface	1
<i>M.F. Goodchild</i> , Introduction: special issue on 'Uncertainty in geographic information systems'	3
<i>P. Fisher</i> , Sorites paradox and vague geographies	7
<i>V. Cross and A. Firat</i> , Fuzzy objects for geographical information systems	19
<i>P.A. Burrough, P.F.M. van Gaans and R.A. MacMillan</i> , High-resolution landform classification using fuzzy k-means	37
<i>D. Scott Mackay and V.B. Robinson</i> , A multiple criteria decision support system for testing integrated environmental models	53
<i>S. Dragicevic and D.J. Marceau</i> , An application of fuzzy logic reasoning for GIS temporal modeling of dynamic processes	69
<i>R.A. MacMillan, W.W. Pettapiece, S.C. Nolan and T.W. Goddard</i> , A generic procedure for automatically segmenting landforms into landform elements using DEMs, heuristic rules and fuzzy logic	81
<i>M.A. Cobb, F.E. Petry and K.B. Shaw</i> , Fuzzy spatial relationship refinements based on minimum bounding rectangle variations	111
<i>H.W. Guesgen and J. Albrecht</i> , Imprecise reasoning in geographic information systems	121
<i>V.B. Robinson</i> , Individual and multipersonal fuzzy spatial relations acquired using human-machine interaction	133
<i>F. Wang</i> , A fuzzy grammar and possibility theory-based natural language user interface for spatial queries	147

CONTENTS
Direct

This journal is part of ContentsDirect, the *free* alerting service which sends tables of contents by e-mail for Elsevier Science books and journals. You can register for ContentsDirect online at: www.elsevier.nl/locate/contentsdirect



0165-0114(20000701)113:1;1-R

- Woods, K., and K. Bowyer. "Computer Detection of Stellate Lesions" in *Digital Mammography*, vol. 1069 of Excerpta Medica International Congress Series. A. Gale, S. Astley, D. Dance, A. Cairns (eds.), Proceedings of the 2nd International Workshop on Digital Mammography, York, England, June 1994. Amsterdam, The Netherlands: Elsevier Science BV, 1994.
- Yin, F. F., M. L. Giger, K. Doi, C. E. Metz, C. J. Vyborny, R. A. Schmidt (1991). "Computer-aided detection of masses in digital mammograms: analysis of bilateral subtraction images." *Med. Phys.* 18: 955-963.
- Zheng, B., Y.-H. Chang, D. Gur (1995) "Computer-aided detection of masses in digitized mammograms using image segmentation and a multi-layer topographic feature analysis." *Acad. Radiol.* 2: 959-966.



Preface

Uncertainty in Geographic Information Systems and Spatial Data

This special issue of Fuzzy Sets and Systems presents a collection of papers on the topic of uncertainty in geographic information systems (GIS) and spatial data. The complexity of spatial data, its interpretation and use in GIS entail many aspects of uncertainty. Our intention for this issue is to more broadly expose researchers in fuzzy logic and soft computing to these topics. They represent a rich environment for further modeling and development by the fuzzy set community. We believe a valuable cross-fertilization with considerable uncertainty research efforts in the GIS community will be facilitated by exposure to papers in this special issue.

The significance of this topic can be assessed by the announcement from the University Consortium for Geographic Information Science (in US) in 1997 that uncertainty is one of the ten research priorities in GIS. These research priorities have had considerable influence already in funding directions. It should also be noted that included among the contributors to this collection are well-known researchers in the area of GIS. Mike Goodchild who has written a special introduction to this collection is a leader in the field and is internationally known for his work on issues of accuracy and imprecision in GIS. Peter Fisher whose article leads off this collection is the editor-in-chief of arguably the most important journal in the GIS area, the International Journal of Geographic Information Science. Another author, Peter Burrough has recently edited an influential volume surveying imprecision concerns for geographic objects.

This special issue has evolved from two special sessions the editors organized at FUZZ-IEEE'98 in

Anchorage, Alaska and IPMU'98 at La Sorbonne in Paris. Several papers from these sessions were extended for this collection and are included as well as others submitted in response to an open call. The papers cover a spectrum of work involving imprecision and uncertainty relative to geographic and spatial concerns. The first two papers in the collection provide a general view of broad issues and representations for spatial data. The next four papers illustrate various usages of fuzzy sets by GIS researchers in the application areas. Then, the following two papers examine some approaches in spatial reasoning based on fuzzy set representation. The last two papers in this volume describe aspects of natural language concerns for uncertainty in GIS.

We wish to thank all researchers who have shown interest in this effort and submitted papers and we also acknowledge the valuable efforts of the reviewers who had to produce their reviews on a very tight schedule. Finally, we gratefully acknowledge the founding editor of Fuzzy Sets and Systems, Hans Zimmermann, who was very supportive and helpful in his handling of this issue.

Maria Cobb
Fred Petry
Vince Robinson

Volume 23 Number 2 May 1999

ISSN 0350-5596

Informatica

**An International Journal of Computing
and Informatics**

Spatial Data Management



The Slovene Society Informatika, Ljubljana, Slovenia

Introduction: Special Issue on Spatial Data Management

Guest Editors:

Frederick E. Petry
 Department of Electrical Engineering & Computer Science
 Tulane University
 New Orleans, LA 70118
 petry@eecs.tulane.edu
 Maria A. Cobb
 Department of Computer Science & Statistics
 University of Southern Mississippi
 Hattiesburg, MS 39406-5106
 maria.cobb@usm.edu
 Kevin B. Shaw
 Digital Mapping, Charting & Geodesy Analysis Program
 Mapping, Charting & Geodesy Branch
 Naval Research Laboratory
 Stennis Space Center, MS 39529
 shaw@nrlssc.navy.mil

This special issue of *Informatica* focuses on several research topics in the area of spatial data management. Spatial databases have developed as extensions to ordinary databases in response to rapidly developing applications such as Geographic Information Systems (GIS), CAD systems, and many multimedia applications. These applications dictate the need for (1) additional data types, including point, line and polygon; (2) spatial operations such as intersection, distance, etc.; and (3) the ability to handle some combination of objects (vector data) and fields (raster data). The nature of spatial data requires multi-dimensional indexing to enhance performance, and much research has been devoted to this topic.

The first set of papers in this issue provides descriptions of extensions to the standard functionality in GIS databases. In particular, these first three papers discuss extensions for space-time visualization, sound as a spatio-temporal field and the inclusion of network facilities in a GIS. A realization of visual aspects of the space-time conceptual framework of Hagerstrand is discussed in the first paper by Hedley, Drew, Arfin and Lee. They demonstrate one of the first examples of an implementation of this approach and provide a real-world case study of visualizing worker exposure to hazardous materials. In the second paper (Laurini, Li, Servigne, Kang) a field-oriented approach to auditory data in a GIS is described. The special semantics of auditory information are presented and some techniques for indexing of such data are indicated. The third paper in this set adds additional levels of abstraction to extend the semantics of GIS networks. The authors, Claramunt and Mainguenaud, then show this

leads to the development of a new interpretation of the database projection operator.

The second set of papers provides techniques for processing and modeling spatial data. The first paper by Havran provides a new approach for memory mapping of binary search trees that can improve spatial locality of data and thus spatial query performance. In the next paper by Chung and Wu, some improved spatial data structure representations including linear quadrees are presented. These representations are shown to have better compression performance. Finally, the paper by Forlizzi and Nardelli describes the lattice completion of a poset to model spatial relationships. They prove some of the needed conditions for valid intersection and union relations among spatial objects with this representation.

Informatica

An International Journal of Computing and Informatics

Introduction		153
Hagerstrand Revisited: Interactive Space-Time Visualizations of Complex Spatial Data	N.R. Hedley C.H. Drew E.A. Arfin A. Lee	155
Modeling an Auditory Urban Database with a Field-Oriented Approach	R. Laurini K.-J. Li S. Servigne M.-A. Kang	169
A Revisited Database Projection Operator for Network Facilities in a GIS	C. Claramunt M. Mainguenaud	187
Analysis of Cache Sensitive Representation for Binary Space Partitioning Trees	V. Havran	203
Improved Representations for Spatial Data Structures and Their Manipulations	K.-L. Chung J.-G. Wu	211
Characterization Results for the Poset Based Representation of Topological Relations - I: Introduction and Models	Luca Forlizzi Enrico Nardelli	223
<hr/>		
Asynchronous Microprocessors	J. Šlic B. Robič	239
Discrete-Event Simulation Software : A Comparison Of Users' Surveys	V. Klupic	249
More Efficient Functionality Decomposition in LOTOS	M. Kapus-Kolar	259
An Application-Level Dependable Technique for Farmer-Worker Parallel Programs	V. De Florio G. Deconinck R. Lauwereins	275
Agent Properties In Multi-Agent Systems	M. Maleković	283
Extended Predicate Logic and Its Application in Designing MKL Language	H. Dai	289
Reports and Announcements		301

Fuzzy spatial relationship refinements based on minimum bounding rectangle variations

Maria A. Cobb^{a,*}, Frederick E. Petry^b, Kevin B. Shaw^c

^aDepartment of Computer Science and Statistics, University of Southern Mississippi, Hattiesburg, MS 39406-5106, USA

^bCenter for Intelligent and Knowledge-Based Systems, Tulane University, New Orleans, LA 70118, USA

^cNaval Research Laboratory, Stennis Space Center, MS 39529, USA

Received November 1998

Abstract

Many spatial data modeling strategies rely upon approximate representations of spatial objects both for computational efficiency issues as well as the simplification of logical modeling strategies. The most widely used approximation is the minimum bounding rectangle (MBR). While the use of MBRs in spatial data modeling is extensive due to their efficiency for storage and relationship calculation, their use as a solitary means of identifying, for example, topological relationships between objects is problematic due to the inconsistency of mappings between relationships of MBRs and corresponding relationships of the objects they represent. In this paper we examine several extensions to the MBR model that reduce the discrepancies between binary spatial relationships of the MBRs and those of the contained objects. For each scheme, we consider the implications to the determination of fuzzy spatial relationships and the impact on computational issues. © 2000 Published by Elsevier Science B.V. All rights reserved.

Keywords: Minimum bounding rectangle; Fuzzy spatial relationship; Conflation; Multiple rectangle representation

1. Introduction

The process of transforming geometric information corresponding to real-world geographic objects into a digital model has almost unlimited potential for the introduction of uncertainty. Particular stages of the process at which a large part of the resulting uncertainty is related include: data capture (due to random errors associated with equipment), discretization (due to sampling resolutions, rounding errors, etc.) and

object identification (due to human-dependent boundary determination and proper labeling). It is possible, in spite of these obstacles, to finally arrive at a digital representation, which, though not perfect, is “good enough”. We mean that some system, such as a geographic information system (GIS), is able to utilize the available data to provide analysis and query results that are perfectly suitable in terms of completeness and accuracy for the end-user or application.

These issues aside, assuming that we have some discrete representation of a geographic domain which is the best possible, yet another issue of uncertainty remains – that of determining relationships among the various geographic objects.

* Corresponding author.

E-mail addresses: maria.cobb@usm.edu (M.A. Cobb); petry@cics.tulane.edu (F.E. Petry); shaw@nrlssc.navy.mil (K.B. Shaw)

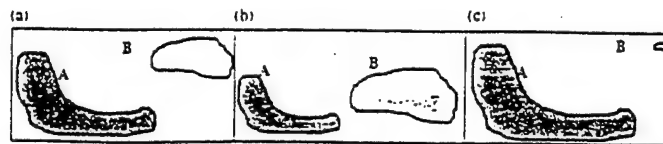


Fig. 1. Example of the effect of size on directional relationship determination.

The ability to discriminate between similar spatial relationships and the ability to communicate subtle differences in such relationships is a difficult task for automated systems. Especially difficult is the determination of directions between 2-D features. The use of fuzzy methods associated with linguistic variables [16] is the most promising approach so far for duplicating the human reasoning process in the area of spatial relationship determination.

As an example, consider the three scenes pictured in Fig. 1. In Fig. 1(a) it is unclear whether the statement “A is west of B” or “A is southwest of B” better describes the directional relationship between the two. In Fig. 1(b), however, it is much less controversial to state simply that “A is west of B”. Similarly, in Fig. 1(c), most would agree that now “A is southwest of B”.

Often, centroids are used to provide a single point of reference for determining orientation of 2-D objects. (e.g. [3]). However, some information loss is inherent in this method. For example, in Fig. 1, south would be the only directional relationship preserved using centroids. The model presented in this paper preserves all of the directional relationships that exist between two 2-D objects, along with a representation indicating the degree to which the objects are determined to participate in each of the relationships. This method of qualifying directional relationships more closely models the way in which humans naturally process and communicate such information.

The following section provides information on the use of minimum bounding rectangles (MBRs) as approximations of geometric objects, including a rationale for their use, associated problems and related work on the support for and derivation of spatial relationships based on their use. This is followed by an overview of the abstract spatial graph model in Section 3. Section 4 follows with descriptions of several alternative MBR approaches. These are analyzed in Section 5 with

respect to geometric modeling capabilities. Application of the various approaches to the issue of conflation is presented in Section 6. Our conclusions regarding the extended approaches are presented in Section 7.

2. Background

The work presented here, as well as that of Nabil [11] and Clementini [4], relies upon the use of MBRs as approximations of the geometry of spatial objects. An MBR is defined as the smallest x - y parallel rectangle that completely encloses an object. The use of MBRs in geographic databases is widely practiced as an efficient way of locating and accessing objects in space [4]. In addition, numerous spatial data structures and indexing techniques have been developed that exploit the computationally efficient representation of spatial objects through the use of MBRs [10,13,15]. Another advantage to the use of an MBR representation is that all objects can be dealt with at the same level of dimensionality – that is, point, line and area features are all represented as 2-D objects across which operations can be uniformly applied.

Clementini [4] shows how MBR relations can be used as a fast filter to determine whether it is possible for the object to satisfy a given topological relationship. This approach is based on the identification of a set of MBR relations for which a consistent mapping between these relations and object topological relations exists. For example, if two MBRs are *disjoint*, then the relationship between the objects must also be *disjoint*. An approach for ameliorating the topological consistency problem is the use of *true* MBRs proposed by Nabil [11]. A true MBR is not restricted by the x - y parallelism requirement, but is designed to represent the true maximum extent of an object unconstrained by orientation. This approach potentially

results in less false area, thereby reducing the margin for error between MBR and object relationship mappings.

Our goals in this research are to (1) determine a representation for spatial objects that will alleviate, or at least ameliorate, some of the problems associated with MBRs, while retaining as many of the desirable characteristics as possible, and (2) show how the resulting representation can be used for modeling fuzzy spatial relationships.

3. Abstract spatial graph model

This section presents a data structure for representing topological and directional relationships, in addition to supplementary information needed for fuzzy query processing. The data structure, known as an *abstract spatial graph* (ASG), represents a transformation of 2-D space (areas) into 0-D space (points). A complete set of ASGs for the original relationships, including a graphical representation and specific property sets, was developed in [5].

First-level topological relationship definitions for these original relationships are based on an extension of Allen's temporal relations [1] to the spatial domain. In this work, Allen showed that the seven relationships *before*, *meets*, *overlaps*, *starts*, *during*, *finishes* and *equal*, along with their inverses, hold as the complete set of relationships between two intervals. Cobb [5] extended these to two dimensions by defining a spatial relationship as a tuple $[r_x, r_y]$ where r_x is the one of Allen's relationships that represents the spatial relationship between two objects in the x direction, and r_y is likewise defined for the y direction. Relationships are often represented by their initial letter; for example, [bo] stands for the relationship [before, overlaps] and $[bo]^{-1}$ represents the corresponding inverse relationship. Inverse relationships imply the reversal of the objects' roles in the relationship. For example, $A [bo] B$ is equivalent to $B [bo]^{-1} A$. Objects involved in any of these relationships are assumed to be enclosed by MBRs.

Formal definitions for each of the relationships are given in terms of a set of constraints based on corner positions, each of which must hold between the MBRs for each object. For example, for the case of A [finishes, starts] B , the definition is given

as: $\{B_{x1} < A_{x1} < B_{x2}, A_{x2} = B_{x2}, B_{y1} < A_{y2} < B_{y2}, A_{y1} = B_{y1}\}$, where $(x1, y1)$ and $(x2, y2)$ represent the lower left and upper right corners, respectively, of the MBRs. This is somewhat similar to the way in which the eight spatial relational operators are defined in [9] by sets of relationships involving the operators *min*, *max* and *length* of objects, which are then combined with object representations to form a 2-D string [2] for spatial reasoning.

The concept of extending Allen's temporal relations to two or more dimensions for spatial reasoning is not new. Examples of how this has been done can be found in [8,11] to name a few. For each, the approach taken is somewhat different, based on the intent of the work. However, the concept of representing a 2-D object as a set of two intervals, an x and a y , and of having the resulting spatial relationship consist of some combination of the component 1-D relations remains key. In contrast, Egenhofer's well-known model for topological relations [7] utilizes a point-set approach in which relationships are based on combinations of intersections between boundaries and interiors of objects.

Directional relationship definitions in [5] rely upon the partitioning of MBRs into object sub-groups, which are created by extending the boundaries of the two MBRs so that they intersect one another. For those cases in which extensions do not intersect the other MBR, each MBR is considered to be an object sub-group. Overlapping areas are also considered object sub-groups. The extensions of the MBR sides that form object subgroups are shown as dotted lines in Fig. 2.

The construction of an ASG for a binary spatial relationship relies heavily upon these object sub-groups. Each object sub-group is represented as a node on the ASG. Pictorially, ASGs are represented in a polar graph notation, where different node representations, e.g., filled vs. open for Fig. 2, are used to distinguish between the objects involved in the relationship. The origin node represents the reference area of the relationship, which could be a sub-group of one of the objects, an overlapping area, or common boundary. An example of an ASG and its corresponding relationship is shown in Fig. 2.

To provide support for fuzzy query processing, each node in an ASG has associated *weights*. These weights are used to define fuzzy qualifiers for the

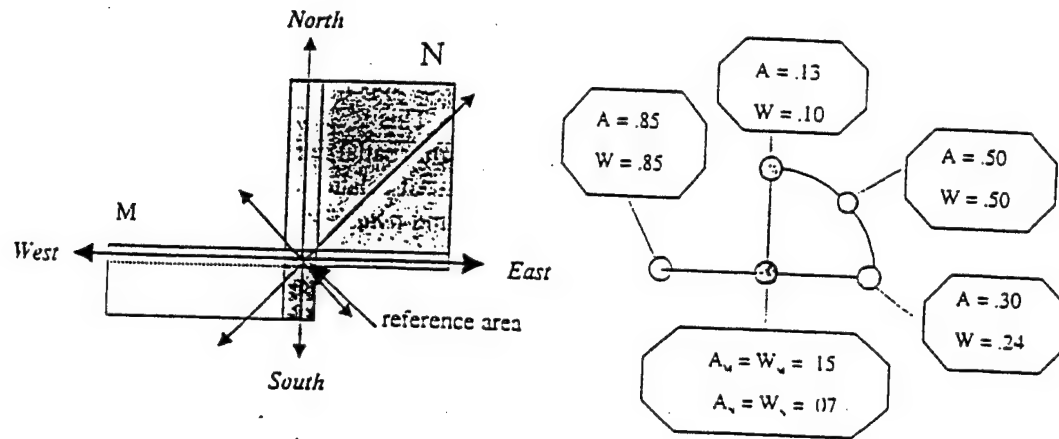


Fig. 2. M [overlaps, starts] N relationship and ASG.

query language. Specifically, the weights are intended to support queries of the nature "To what degree is region *A* south of region *B*?", or "How much of region *A* overlaps region *B* (qualitatively speaking)?"

Two types of weights are computed: area weights and total node weights, which are represented in Fig. 2 as *A* and *W*, respectively. These provide information concerning the degree of participation in a relationship and relative direction, respectively. Area weights are computed as the ratio of the area of an object sub-group to the area of the entire MBR. The total node weight for an ASG node is defined as the product of the corresponding area weight and the normalized axis length of the directional axis which crosses the object sub-group. Normalization of axis lengths is accomplished for each object by first assigning a length of 1 to the longest axis crossing any object sub-group of the object. All other representational axes of the object are then given a value between 0 and 1, based on their lengths relative to the longest axis for that object.

From the definition, one can see that area weights are useful for answering *how much* of an object is involved in a relationship. By assigning ranges of area weights to linguistic terms we can provide a basis for processing queries concerning qualitatively defined relationships. The set given below is one example of how this may be done.

{all (96–100%), most (60–95%), some (30–59%), little (6–29%), none (0–5%)}

This provides the capability to pose queries such as the following:

- Is object *A* surrounded-by most of object *B*?
- Retrieve an object that is partially-surrounded-by little of object *A*.

Node weights are utilized in a similar manner to provide qualitative directional relationship information. The purpose of node weights is to answer the *extent* to which an object can be considered to be at a given direction in relation to another object. The definition of a node weight as the product of the area weight and the axis length means that this information is represented as a consideration of both the total relative amount of the object which lies in a given direction (represented by the area weight) tempered by how *directly* it lies in that direction (represented by the axis length). This implies that those directions which have both a large area representation and long axis length will have a higher weight than those which have either a large area representation but short axis length, or those that have a long axis length with lesser area representation. Again, ranges are provided that define a linguistic set useful for query purposes. These are given below.

{directly (96–100%), mostly (60–95%), slightly (30–59%), somewhat (6–29%), not (0–5%)}

The use of these qualifiers is illustrated in the following:

- Is object *B* somewhat west of object *A*?
 - Retrieve an object directly northeast of object *A*.
- The preservation of all directional information regarding two objects, along with the use of total node

weights as described here, allows users to obtain a complete conceptualization of directional relationships between the objects. Furthermore, the calculation of the total node weight as the product of the axis weight and area weight ensures against bias in those cases, for example, where the object sub-group associated with a directional axis is very large (increasing the weight for that direction), but for which the axis weight is very small (indicating a weaker association for that sub-group, direction pair than for others for the same object).

4. Extensions to the standard MBR representation

The ASG model utilizes MBRs to represent features for the advantages given in the preceding section, while algorithms for computing relationships for the model are designed in such a way as to minimize the potentially negative impact of the use of MBRs. Additionally, the ASG model extends the use of rectangular boundaries as representations of sub-objects within the MBRs.

Of course, the use of MBRs is inherently problematic to some degree because an MBR is an approximation of an object's true geometry. One of the more significant challenges is the modeling of topological spatial relations using MBR representations. The problem is that the enclosure of false area (area not actually contained in the geometry of the object) within the MBR renders inconsistencies between the application of topological relations to the MBRs vs. the application of the relations to the objects themselves. Such an inconsistency is especially evident in the case of overlapping relationships for which the MBRs may overlap, but for which no conclusion can be drawn about the corresponding relationship of their respective contained objects.

In keeping with generally accepted practice, both MBRs and the sub-rectangles utilized in the ASG model retain *x-y* parallelism; however, in this section we explore several variations on this traditional approach and investigate the respective implications to the modeling of relationships. Our goal in investigating alternative representations for geometric properties of spatial features is threefold: (1) alleviate or significantly decrease anomalies in topological relationship determination based on MBRs; (2) im-

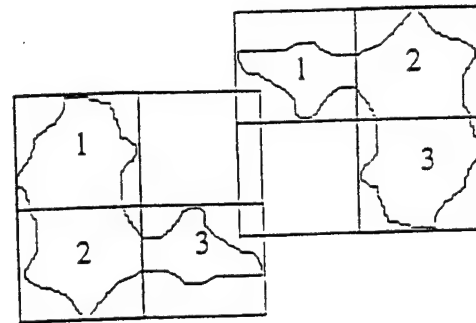


Fig. 3. Uniform MRRs used for object representations.

prove accuracy in determination of directional and topological relationships between representations; and (3) maintain, as much as possible, computational efficiency in processing concerning relationship categorization and querying.

First, we consider the implications of partitioning MBRs into sets of rectangles, essentially resulting in a gridded surface which is an approximation we call *multiple rectangle representation*, or MRR. Three variations on MRRs which we analyze in this section include (1) a uniform MRR, (2) a non-uniform, congruent MRR, and (3) a non-uniform, non-congruent (irregular) MRR. All three variations of MRRs result in a finer approximation of the object's true geometry than do MBRs, while maintaining a basic regular, rectangular structure for which computationally efficient methodologies have been developed.

The first variation can be viewed as the imposition of a grid of any level of resolution upon an object, after which any of the rectangles not actually intersecting with a part of the object is discarded. Two cases presented here include one in which grids of the same resolution are used for both objects participating in a relationship, as well as the case in which grids of different resolutions are used. Fig. 3 shows a simple example of the use of uniform MRRs. The dotted line shows the original boundaries. Numbers for grid cells have been assigned arbitrarily, and are used in following discussions on relationship definitions.

Now, we consider enhancements to the ASG to accommodate the use of MRRs for relationship determination. We begin with the supposition that the set

of ASGs is a closed set, such that any modification to the way in which relationships are defined does not result in any new ASG. This issue is discussed further in Section 5. It is apparent, however, that the way in which the ASGs themselves are defined for MRRs must be modified to take advantage of the more accurate representation.

We do this by first computing a set of ASGs – one ASG for every combination of relationships between sub-rectangles of both objects' MRRs. This results in a set of ASGs for each binary relationship. $S = \{A_{ij}, 1 \leq i \leq n, 1 \leq j \leq m, n = \text{number of subrectangles in first object}, m = \text{number of subrectangles in second object}\}$. For example, in Fig. 3 are two objects with simple, uniform MRRs. The resulting set of relationship ASGs for this example is

$$\begin{aligned} S = \{ & A_{11} = [\text{bo}], A_{12} = [\text{bo}], A_{13} = [\text{bo}^{-1}], \\ & A_{21} = [\text{bb}], A_{22} = [\text{bb}], A_{23} = [\text{bo}], \\ & A_{31} = [\text{ob}], A_{32} = [\text{bb}], A_{33} = [\text{bo}]\}. \end{aligned}$$

An approach for utilizing this information set to gain a more accurate picture of the relationship under consideration is to first associate a count with every distinct relationship appearing in the set. For our example, this would yield $S' = \{([\text{bo}], 4), ([\text{bb}], 3), ([\text{bo}^{-1}], 1), ([\text{ob}], 1)\}$. These counts divided by the total number of sub-rectangle relationship combinations (9 in the example) are considered as membership values in fuzzy relationships. Rather than having a single, crisp binary relationship as was the case in the original ASG model, we now have a set of ASG relationships, each member of which a given binary relationship belongs to some degree. For example, the fact that [bo] and [bb] appear as the predominant relationships for Fig. 3, along with the fact that [bo⁻¹] and [ob] exist, although to a much lesser degree, conveys a substantially more significant amount of information than does the original [oo] designation, which in this case is also inaccurate with respect to the contained objects' relationship due to the topological inconsistency problem associated with MBRs.

We then take this a step further by associating these membership values with the higher-level relationships defined in [5]. Because these relationships represent a mutually exclusive, total partitioning of the basic

relationships, a mapping from the members of S to these relationships will result in at most the same number of relationships as the number of members of S . However, it is more often the case that fewer high-level relationships result. This is because (1) such relationship definition sets are often composed of basic relationship neighbors, and (2) the use of MRRs in the manner described necessarily implies relationship categorizations for neighboring sub-rectangles, resulting in neighboring relationships. In our example, each of [bo], [bb], [bo⁻¹], and [ob] appears in the *disjoint* relationship, leading to the invariable conclusion that the two representations are indeed disjoint.

This approach eliminates the need to compute the set of weights for ASG nodes as was done in the original model, as similar information (the counts are analogous to the original weights) is now maintained in S' .

The application of non-uniform, congruent MRRs can be understood as an analog to a quadtree decomposition commonly performed for spatial indexing purposes. The approach begins with standard x - y parallel MBRs, upon which a quadtree-like decomposition is performed with the MBR being divided into four equivalent rectangles, any or all of which can then be divided similarly, continuing until as fine a partitioning as desired is achieved. At that point, any rectangles not actually containing a part of the object are discarded. We say the rectangle sets are *regularly hierarchical* because the level of detail (size) of the rectangles can vary across different parts of the object so as to achieve a desired level of representation, while each larger rectangle is exactly a multiple to the fourth of any of the smaller rectangles of the object. An example of this type of MRR is shown by the grayed area in Fig. 4.

This type of boundary approximation more accurately represents the objects' geometry in comparison to either MBR or uniform MRR representations. While maintaining the greatly reduced incidence of topological inconsistencies between true and approximate boundaries illustrated in the uniform MRRs, additional levels of detail have been added that allow for more accurate relationship determination. Since the areas are still rectangular decompositions, computational issues remain simplified compared to boundary representations.

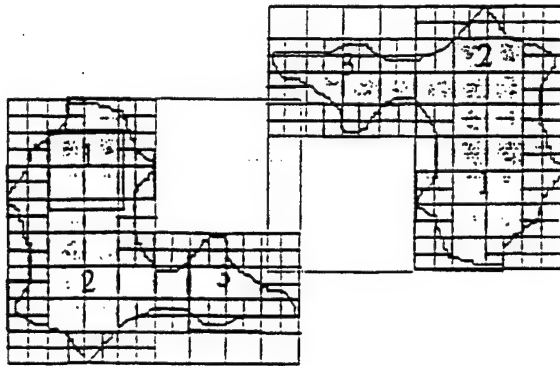
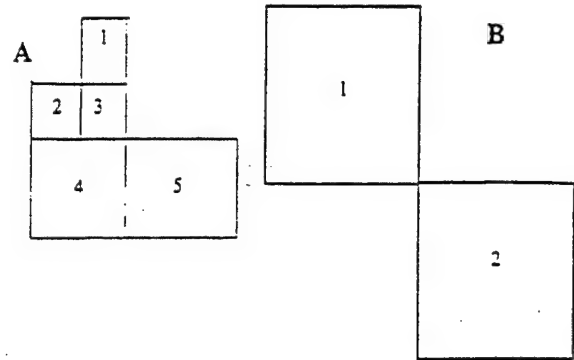


Fig. 4. A non-uniform congruent MRR.

The most significant issue that must be addressed concerning fuzzy relationship modeling for this approach is the manner in which the differently sized rectangles are assessed as contributors to overall relationship determination. This can be handled by extending the approach used for uniform MRRs. While for uniform MRRs it was sufficient to use the mere existence of the basic relationships between sub-rectangles as equal factors in fuzzy relationship categorizations, we must now compute a weight analogous to the ASG node weights for each sub-rectangle relationship to achieve a level of normalization for “combining” the individual relationships into one.

Recall that the area weights are computed as the ratio of the MBR sub-object areas to the total MBR area, and that these weights are used to identify the extent to which an object participates in a given relationship. Using this same approach for non-uniform congruent MRRs – calculating a weight as the ratio of a sub-rectangle’s area to the combined area of all sub-rectangles containing a part of the object – we achieve a level of normalization for use of differently sized rectangles.

For each applicable relationship, the weights for every different sub-rectangle of each object for that relationship are summed, resulting in a value ≤ 1 . Whenever one sub-rectangle participates in the same relationship with multiple sub-rectangles of the second object, that sub-rectangle’s weight is only counted once. The resulting sums for a relationship for the two objects are then multiplied, yielding a weight for that relationship. For example, consider the case illustrated in Fig. 5.



$$A1_w = .1; A2_w = .1; A3_w = .1; A4_w = .35; A5_w = .35$$

$$B1_w = .5; B2_w = .5$$

Fig. 5. Non-congruent MRRs with area weights.

For this example, we have the following set of relationships:

A/B	1	2
1	[bd]	[bb ⁻¹]
2	[bd]	[bb ⁻¹]
3	[bd]	[bb ⁻¹]
4	[bo]	[bo ⁻¹]
5	[bo]	[bo ⁻¹]

By summing and multiplying the area weights in the manner described earlier, we obtain the following:

$$[bd]_w = 0.15, \quad [bo]_w = 0.35,$$

$$[bb^{-1}]_w = 0.15, \quad [bo^{-1}]_w = 0.35.$$

This shows that the relationships [bo] and [bo⁻¹] are weighted more heavily, primarily due to the larger areas of sub-rectangles 4 and 5 of object A. These weights can then be associated with the higher level relationships in the same manner as was shown for the uniform MRRs.

One final possibility for the use of a multiple rectangle representation includes the application of irregular rectangles – meaning any ad hoc method of partitioning the object into rectangular sets such that there is no apparent relationship between the rectangles. This could be done, for example, by a human operator in order to achieve some objective such as finding the “best”

rectangular coverage of an object, or minimizing either the number of rectangles used or the false area within rectangles.

While this may appear to be a special case, careful consideration will show that the same method given in the section for non-uniform congruent MRRs is equally applicable for irregular MRRs.

5. Geometric modelling capabilities

The use of MRRs for the ASG model has several implications. First is that the partitioning of MBRs in the ways described has *no effect* on the basic relationship definitions. Any relationship that originally held between two MBRs remains valid for the resulting MRRs, because minima and maxima for the objects do not change; therefore, any partitioning of the MBRs, regardless of granularity, will not affect the basic relationship between the geometric representations of the objects.

To support this statement, we begin by observing that the abstract spatial graphs can be arranged in a matrix according to the concept of *conceptual neighborhoods* with respect to the relationships that are represented. That is, each of the relationships that borders a particular relationship in both the horizontal and vertical directions can be derived from the given relationship without transitioning through any other relationship state. If a transition to an immediate neighboring relationship is impossible, then it follows that a transition to any other relationship is also impossible. We illustrate this by first examining the example shown in Fig. 6, which represents a portion of the complete relationship matrix.

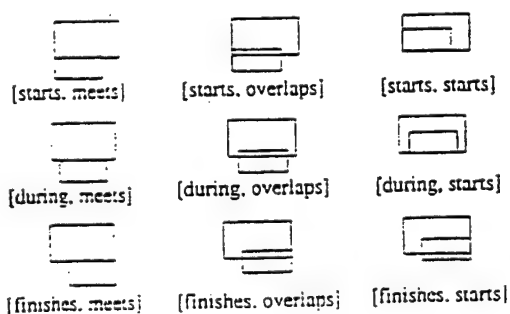


Fig. 6. [during, overlaps] relationship and conceptual neighbors.

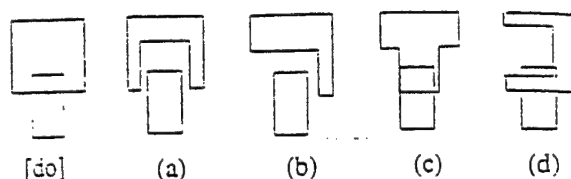


Fig. 7. A set of geometric representations for the [do] relationship.

Now, we show that the enhanced accuracy of the boundaries for the MRR approximation model provides additional information in the way of geometric refinements for the relationship definitions. While there originally was only one geometric model for each relationship, there is now an infinite set of such models that correlates to any given relationship. Simple examples for a standard MRR approach are shown in Fig. 7. Internal rectangle boundaries are omitted for clarity.

From this illustration one can see that any selected MRR partitioning for a pair of MBRs will never cause a transition to one of the neighboring relationships. Therefore, we are able to operate under the assumption of a closed set of relationships for which the ASG model and query framework hold.

Fig. 7 also demonstrates the advantages associated with the use of MRRs over MBRs: (1) the ability to better represent correct topology, and (2) the ability to make finer distinctions between geometric relationships. The first of these is illustrated in Fig. 7 (a), which shows that the two objects could not possibly overlap, while the original MBR representation shows an overlap. The second advantage can be seen Fig. 7(a–b) and (c–d) which show the same topological relationships – disjoint and overlaps – but for which geometric distinctions exist which provide additional information for spatial analysis.

6. Application to conflation

In this section, we will consider the application of these approaches to the problem of conflation. Conflation is typically regarded as the combination of information from two digital maps to produce a third map that is better than either of its component sources. The history of map conflation goes back to the early to mid-1980s. The first clear development and application of an automated conflation process oc-

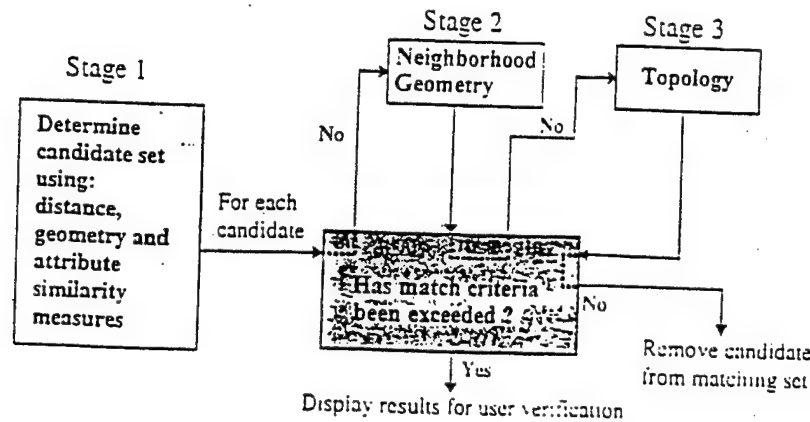


Fig. 8. Conflation process.

curred during a joint United States Geological Survey (USGS)-Bureau of the Census project designed to consolidate the agencies' respective digital map files of US metropolitan areas [12]. The implementation of a computerized system for this task provided an essential foundation for much of the theory and many of the techniques used today. Since that time, others, including commercial GIS vendors, have implemented conflation tools within their applications. For an example of commercial work on conflation, see [14].

Automated conflation of maps is a complex process that must utilize work from a wide range of subjects, including pattern recognition, statistical and graph theory, and a collection of heuristics that hardcopy cartographers have used for decades to enhance aesthetics and increase information content of paper maps.

Conflation of maps typically is needed because: (1) users wish to update their mapping information without losing legacy data which may not be included in the new information; (2) one map source may be more accurate with respect to, e.g., positional or attribute information; or (3) one map source contains information missing in another, such as additional features, feature attributes or even entire coverages.

Conflation can, in brief, be viewed as a multi-step, iterative process that involves feature matching, positional re-alignment of component maps and attribute deconflation of positively identified feature matches, as seen in Fig. 8.

This process was first presented in [6]. As can be seen from the figure, the process begins at stage 1 by finding a candidate set of matches for a feature

from those in the complementary database that are geographically close. The "closest" feature and the one being matched are then compared according to their representation type (point, line or area) and attribute and value sets.

Stage 2 continues with a more detailed analysis of the feature pair. Selected neighbors and their relative positions are investigated. Similarities or dissimilarities are considered as part of the evidence for or against the hypothesis of equivalent features. Considering this first for entity nodes (points which do not represent the intersection of edges), we must investigate properties such as absolute position, positions (distance and direction) relative to nearby features that have already been determined to be matching features, as well as similarities in general neighborhood patterns. For connected nodes (nodes that represent the intersections of edges), geometric considerations include the directions and lengths of intersecting edges (including the spider function). Orientation and length of line features, as well as similarity in size and position of bounding boxes can be used. Similar criteria are used for area features, in addition to the equivalence of values representing the contained area of each.

The final stage analyses the topology of the two features at the lowest level of topology supported by the component databases. For products utilizing winged-edge topology, topological connectives of matching feature candidates can be checked for similarity. This can include, for example, left and right edges and left and right (adjacent) faces of edges for line and area features, or the containing face of point features.

Feature matching, simply and perhaps somewhat obviously stated, involves the identification of features from different maps as being representations of the same geographic entity. Positional alignment is a mathematical procedure in which previously identified matching features are brought into spatial agreement, while deconfliction is a process in which contradictions in a matching pair's attributes and/or values are resolved. Positional alignment and deconfliction are both steps that are performed after a positive match has been determined. As such, it is easy to see that accurate feature matching results are essential to the overall quality of the resulting conflated map.

Two aspects of feature matching in the conflation process can be improved by the representation techniques we have discussed in this paper. First, the use of an MRR will permit more accurate matching of geometric properties of features than MBRs; however, the computational burden of the matching will not be excessive as in direct boundary representation techniques. Thus, MRRs used in this manner provide a filtering mechanism that can be refined as needed based on conflation requirements. Secondly, the refinements of topological relationships, such as we have illustrated in Fig. 7, allow the third stage of the feature matching process to be enhanced. It is clear that there is an improvement over the simple use of MBRs, as illustrated in Fig. 6, but still again, it does not require a large computational overhead demanded by formal topological calculations.

7. Summary and conclusions

In this paper we have presented several alternative approaches for approximate geometric boundary representation based on both standard and true MBRs. These include variations on MRRs for uniform, non-uniform but congruent, and irregular rectangular decompositions. For the standard MBR-based approaches, we have shown how the abstract spatial graph model for fuzzy spatial relationships could be extended to accommodate the new representations.

From this work, we believe the most promising approaches lie with the non-uniform but congruent and the irregular MRRs. These two offer the best trade-off in terms of increased representational accuracy and maintenance of reduced computational complex-

ity. For future work, we intend to provide a quantitative analysis of the various representations based on discrimination capabilities and computational overhead, as well as determine the implication for directional relationships.

References

- [1] J.F. Allen, Maintaining knowledge about temporal intervals, *Comm. ACM* 26(11) (1983) 832–843.
- [2] S.K. Chang, Q.Y. Shi, C.W. Yan, Iconic indexing by 2D strings, *IEEE Trans. Pattern Recognition Mach. Intell.* 9(3) (1987) 413–428.
- [3] S.K. Chang et al., An intelligent image database system, *IEEE Trans. Software Eng.* 14(5) (1988) 681–688.
- [4] E. Clementini, J. Sharma, M.J. Egenhofer, Modelling topological and spatial relations: strategies for query processing, *Comp. Graphics* 18(6) (1994) 815–822.
- [5] M. Cobb, An approach for the definition, representation and querying of binary topological and directional relationships between two-dimensional objects, Ph.D. Thesis, Tulane University, 1995.
- [6] M. Cobb, M. Chung, V. Miller, H. Foley III, F. Petry, K. Shaw, A rule-based approach for the conflation of attributed vector data, *Geoinformatica* 2(1) (1998a) 7–35.
- [7] M.J. Egenhofer, Spatial query languages, Ph.D. Thesis, University of Maine, 1989.
- [8] H.W. Guesgen, Spatial reasoning based on Allen's temporal logic, Technical Report TR-89-049, International Computer Science Institute, Berkeley, CA, 1989.
- [9] E. Jungert, Extended symbolic projection used in a knowledge structure for spatial reasoning, 4th BPRA Conf. on Pattern Recognition, 1988, pp. 343–351.
- [10] H.P. Kriegel, M. Schiwietz, R. Schneider, B. Seeger, Performance comparison of point and spatial access methods, in: A. Buchmann, O. Gunther, T. Smith, Y. Wang (Eds.), *Design and Implementation of Large Spatial Databases*, Lecture Notes in Computer Science, vol. 409, Springer, Santa Barbara, CA, 1989, pp. 89–114.
- [11] M. Nabil, J. Shepherd, A.H.H. Ngu, 2D projection interval relationships: a symbolic representation of spatial relationships, *Advances in Spatial Databases: 4th Internat. Symp. SSD '95*, 1995, pp. 292–309.
- [12] A. Saalfeld, Conflation: automated map compilation, *Internat. J. GIS* 2(3) (1988) 217–228.
- [13] H. Samet, *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*, Addison-Wesley, Reading, MA, 1989.
- [14] D. Siegel, A non-traditional solution to conflation, *Proc. Urban and Regional Information System Association Annual Conf.*, Washington, DC, 16–20 July, 1995, pp. 462–75.
- [15] V. Gaede, O. Gunther, Multidimensional Access Methods, *ACM Comp. Surveys* 30(2) (1998) 170–231.
- [16] L. Zadeh, The concept of a linguistic variable and its application to approximate reasoning, *Information Sci.* 8(3) (1975) 199–249.

FUZZY

sets and systems

International Journal of Soft Computing and Intelligence

official publication of the
International Fuzzy Systems Association

IFSA

Volume 113 Number 1 July 1, 2000

Special Issue

Uncertainty in Geographic Information Systems
and Spatial Data

Your First-Stop Source for
Computational
Intelligence Information
and Services on the Web
www.elsevier.com/cite

COMPUTATIONAL
INTELLIGENCE
CITE

north-holland



Querying Multiple Data Sources via an Object-Oriented Spatial Query Interface and Framework

MYI CHUNG*, RUTH WILSON*, KEVIN SHAW*, FREDERICK E. PETRY[†]
AND MARIA A. COBB[‡]

**Naval Research Laboratory, Stennis Space Center, MS 39529, U.S.A. E-mail: chung.ruth.wilson, shaw@nrlssc.navy.mil; †Electrical Engineering and Computer Science Department, Tulane University, New Orleans, LA 70118, U.S.A. E-mail: petry@eecs.tulane.edu and ‡Department of Computer Science and Statistics, University of Southern Mississippi, Hattiesburg, MS 39406-5106, U.S.A. E-mail: maria.cobb@usm.edu*

Received 30 March 2000; revised 1 August 2000; accepted 30 August 2000

A spatial query interface has been designed and implemented in the object-oriented paradigm for heterogeneous data sets. The object-oriented approach presented is shown to be highly suitable for querying typical multiple heterogeneous sources of spatial data. The spatial query model takes into consideration two common components of spatial data: spatial location and attributes. Spatial location allows users to specify an area or a region of interest, also known as a spatial range query. Also, the spatial query allows users to query spatial orientation and relationships (geometric and topological relationships) among other spatial data within the selected area or region. Queries on the properties and values of attributes provide more detailed non-spatial characteristics of spatial data. A query model specific to spatial data involves exploitation of both spatial and attribute components. This paper presents a conceptual spatial query model of heterogeneous data sets based on the object-oriented data model used in the geospatial information distribution system (GIDS).

© 2001 Academic Press

Keywords: object-oriented technology, spatial query, data integration, quadtree, object-oriented spatial query, digital mapping; databases.

1. Introduction

A QUERY IS AN INTERACTION between an end-user and one or more databases. The user formulates a request based on what she/he wants to know. Therefore, without having in-depth knowledge of the underlying data, many users are able to ask questions pertinent to their work. A large volume of data is available in many different formats. Users expect to retrieve all relevant information from multiple data sources. Users then determine the usefulness of the data based on the quality of the response and the usability of the response to a query. Query processing is impacted by the ability to abstract data as well as by the language itself. Therefore, both a data model and the query language need to be considered in the context of query processing.

Spatial data are complex data types. Spatially related data have two components: (1) location and its extensions, and (2) attribute data describing non-spatial properties.

Spatial data can be referenced by specific geographic space. Attributes of spatial data provide information regarding those characteristics that are not necessarily spatial; they provide a detailed description of the spatial data by supplying values for its properties. In the typical environment described in this paper, spatial data from multiple sources and formats (i.e. coverages, scales, etc.) are accessed and integrated to provide query responses. The object-oriented query approach we will describe is well-suited for providing the mechanisms needed in such a diverse heterogeneous environment.

Egenhofer defines two fundamental concepts of spatial data abstraction: the view of spatial data in the form of complex objects, and the interaction with spatial objects through pertinent operations [1]. The geographic community has accepted the object-oriented modeling of spatial data as a natural paradigm for geographic objects [2]. The data encapsulation property of an object-oriented model allows an aggregate of objects to be considered as a unique, comprehensive object. Thus, there is a greater ability to localize a query to a subset of objects. An object model is simplified compared to the relational model since join conditions need not be specified. Another difference from the relational data model is that pertinent behaviors are encapsulated with the data. Therefore, an object is a completely self-contained module of state and behavior. Direct queries and subsequently directed retrieval can be performed on objects. Query processing of an object extends the traditional extraction of state information to include behaviors of the spatial object. Thus, query processing of objects involves more than a simple retrieval of stored information; it may initiate some computation as a behavior of an object and yield a query response as a result.

This paper will present a spatial query description to support spatial query processing of an object-oriented spatial data model. This formal method provides the basis for articulating the issues and approaches in querying multiple sources of spatial data. Section 2 introduces issues and background on the integration of heterogeneous spatial data types as related to the query model, followed by Section 3 on discussion of the spatial query language issues. Concerns such as the use of an object-oriented methodology and its implications for querying are discussed. Section 4 addresses a generalized formal description of the object-oriented spatial query model (oo-sqm). Section 5 provides insight into the implementation issues of the spatial query model for a geospatial information distribution system (GIDS), followed by our conclusions and directions for future work in Section 6.

2. Heterogeneous Data Integration and Query Model

Three types of query processing must be considered for spatial data: spatial, attribute, and a combination of spatial and attribute querying. A spatial data search always requires some reference to the location or an area of interest (AOI). A spatial query model must consider access patterns of spatial data. In general, data access begins from a general neighborhood search to a specific location. From such general access patterns, both hierarchical and geographic clustering mechanisms or indexing yield favorable results. A comprehensive study on different spatial access mechanisms is presented in Gaede [3] with a detailed study on spatial data structures presented in Samet [4].

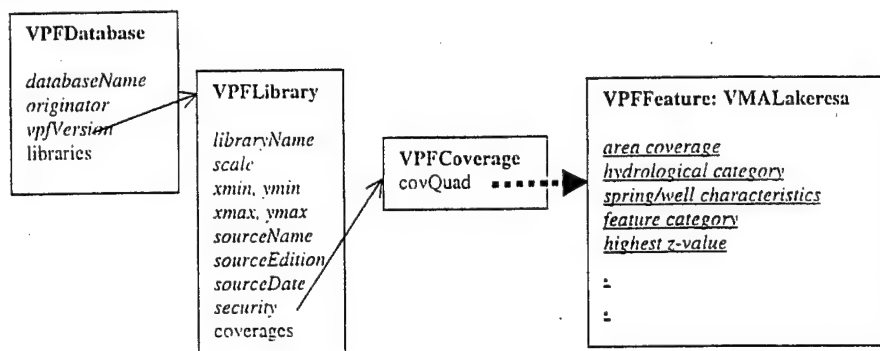


Figure 1. An example of meta-data and spatial data attributes

The non-spatial details of spatial data are considered as a part of attribute-level querying. Usually, a distinction is made between data regarding the spatial information and data related to the actual description of spatial data (metadata), as shown in Figure 1. In this figure, an example of one data type used in the GIDS, namely vector product format (VPF) data, is used to show the difference between metadata and feature attributes. Information in italics are metadata that describe the form of the actual spatial data contained in the data set. The italicized and underlined description provides specific information regarding the characteristics and properties of the spatial data; these are the attributes of the spatial data. However, in the heterogeneous data integration model, both the metadata and spatial object description are considered as attributes. For example, in the figure an instance of the class **VPFLibrary** contains metadata relative to the scale at which the data were collected, as well as the security classification of the contained data, while the feature class **VMALakeresa** contains an attribute *hydrological category* that is not considered to be metadata. Due to the different level of data abstraction for the meta-data and the spatial data, a knowledge-based system that governs the attribute evaluation, especially for the metadata, is one consideration to be made. An attribute-based query may be as simple as a question on one of the attribute types or a value, or it can be as complicated as a sum of some products of attribute type or values.

In the real world, there is only one representation of the spatial object at a given location. However, because of multiple data collections and varying scopes and intentions of data users, many different representations of the real world may exist in a digital environment. In other words, based on the collection criteria, and specifically the resolution at which the data were collected, a single real-world object may be modeled differently in different data sets. For example, at a high resolution, a building may be represented as a polygon of its footprint. But, at a low resolution, the same building may be represented as a point feature. Thus, in spatial data modeling, the resolution and scale at which the data are represented becomes important, especially in a multi-data source and format environment.

Spatial data are closely tied to their graphical representations. Therefore, when multiple data types from data sources are requested to be queried simultaneously, the integrity on the similarity of graphical representation must be insured. This can be achieved by allowing the user to specify a scale. Even though there may be spatial

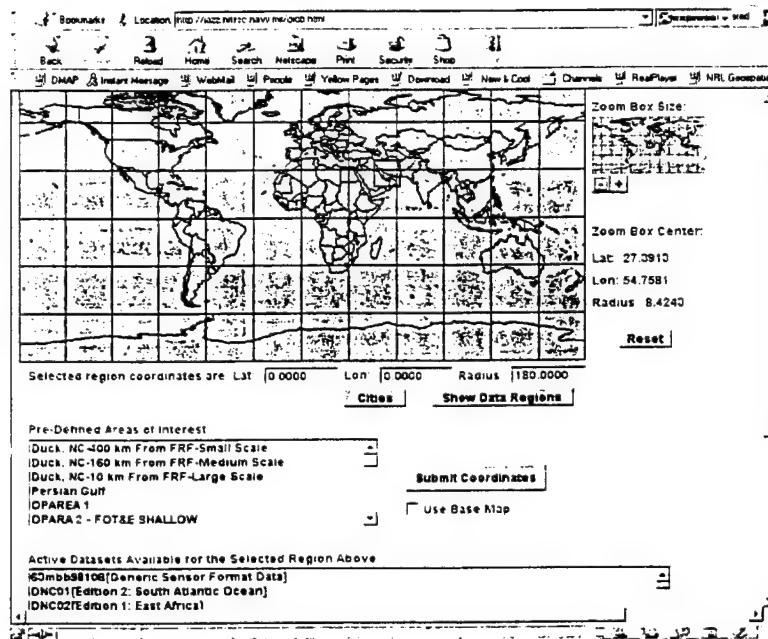


Figure 2. AOI and related datasets

objects that meet the attribute criteria, the collection criteria may have been different. The scale issue can be considered as a part of attributes if some level of a conflation process as discussed in Cobb [5] is implemented. Until a knowledge-based system that can handle the feature matching and feature representation at multiple scales by applying generalization is developed, the inclusion of scale in a query model implementation is not feasible. However, the significance and relevance of scale to the issue of spatial queries is undeniable, and is thus included in the theoretical model and immediately following discussion. It follows from this presentation, then, that a spatial query is a function of at minimum three criteria: geo-location or an area-of-interest (AOI), scale, and attributes. Therefore, we define a spatial query model as

$$SQM = f[AOI, scale, \Sigma attributes].$$

Due to the multi-resolution issue in multi-datasource databases, the outcome of the SQM should be a conjunction of the three parameters: $AOI \wedge scale \wedge \Sigma attributes$. This integration of queries based on AOI is illustrated in Figure 2. This figure shows the area selected by the rectangle on the map (Persian Gulf region), as well as the active datasets for that region, i.e. all member datasets that contain data in any format for the region. The figure shows several such datasets, including VPF (DNC01 and 02) and Generic Sensor Format data. The AOI is a critical component of the query model that is used at a high level to determine the datasets over which the remaining query constraints are evaluated.

Within each parameter, such as the attribute constraints, a combination of logical operators (AND, OR, and NOT) can be used. A query formulated in this manner can

have a number of constraint issues relative to the combinations of parameters. For example, what should be the priority of evaluation of the query expression with respect to the three operators? This ordering can clearly affect retrieval efficiency and require accessing of more or fewer data sources depending upon the evaluation order.

Another consideration might be an absolute requirement on one or more of the query parameters. For instance, if a given scale is set as an absolute requirement, this could dictate a query failure unless the *-AOI* could be changed, e.g. reduced. If, however, a range of scale is specified, a careful evaluation of the scale range and the attribute constraints needs to be considered. A range of scale can be considered as a disjunction of each scale ($scale_i \vee scale_j$). Then, using \wedge as AND operator and \vee as OR operator,

$$(-AOI \wedge (scale_i \vee scale_j) \wedge \Sigma \text{ attributes}) = (-AOI \wedge scale_i \wedge \Sigma \text{ attributes}) \vee (-AOI \wedge scale_j \wedge \Sigma \text{ attributes}).$$

In this context of graphical representation, we may have integrated issues of scale and resolution. There may be some spatial features, f_i , that are collected at $scale_i$ that meet the *-AOI* and *attribute* constraints. There are some other spatial features, f_j , that are collected at $scale_j$ that also meet the *-AOI* and *attribute* constraints. Due to the differences in the scale, integrated display of both feature sets would produce a mis-representation of the spatial features. Thus, a requirement of the *SQM* is that query results must contain spatial features at the same scale. Applying a generalization function that maps those spatial data collected at a high resolution to the low resolution enables a consistent graphical representation of both spatial feature sets. Assuming $scale_j$ is at a higher resolution than $scale_i$, the *SQM* result is

$$(-AOI \wedge (scale_i \vee scale_j) \wedge \Sigma \text{ attributes}) = (-AOI \wedge scale_i \wedge \Sigma \text{ attributes}) \wedge Gen_j \rightarrow (-AOI \wedge scale_j \wedge \Sigma \text{ attributes}).$$

The generalization function will only modify the spatial component of the spatial object. The attributes of the spatial objects will not be affected. The non-spatial description of the spatial object will be at a higher resolution. However, this does not violate the integrity of the geographical information. The implication is that the generalization function will be applied only when the data are requested to be displayed.

Likewise, a class hierarchy that exists in the object-oriented data model would allow attribute abstraction from the superclass level. Due to the inheritance property of the object-oriented data model, the attributes or the properties of the superclass are also defined for subclasses.

In summary, an *SQM* as defined provides a way to uniquely abstract spatial features by specifying the three minimum search criteria. An AND function is imposed among the three constraints. However, in the case where a range of scale is specified, then all the spatial features that meet the scale range and the attribute constraints are retrieved. In this case, though, a generalization function must be applied to the spatial features

collected at the higher resolution to display features of different scales in a consistent manner.

3. Spatial Query Language Issues

3.1. Overview of Spatial Query Approaches

In this section, we survey various spatial query approaches and then provide an assessment related to our querying approach. Spatial query languages are by nature more complex than their alphanumeric counterparts. Additional requirements for spatial query languages, including the ability to handle queries containing both spatial and non-spatial selection criteria, as well as providing appropriate contextual graphical display of spatial query results and intuitive user interface functionality (e.g. point-and-click), greatly complicate the design and implementation of spatial query languages. Applications of spatial query languages vary widely and include areas such as geographical information systems (GIS), pictorial and image and general multimedia databases and CAD/CAM systems. There are several ways to view the various approaches to spatial query languages. In particular, here a very important issue is the distinction between languages based on relational vs. object-oriented models. Also to be considered is the relevant domain, in particular whether the focus is strictly on spatial data or more generally multimedia, often including images, audio, video and subsuming some aspects of spatial data also.

First, for relational model-based querying, we must consider those languages based on SQL. Spatial SQL [6] consists of two separable components: a query language for information retrieval, and a presentation language for directing the display of spatial data. This separation of querying was intended to reduce the complexity of the query structure for users. The graphical presentation language (GPL) is a superset of the query language spatial SQL, which is an extension of SQL with spatial data handling capabilities. It provides a new domain—spatial—with specializations for points, lines, area and 3-D components. Spatial operators and spatial and topological relationships are supported and can be used in predicates of the WHERE clause of an SQL statement. PSQL [7] was developed for pictorial database systems. Points, lines and regions comprise the pictorial domain of PSQL. The language was designed to be flexible, allowing for user-defined operators and abstract data types, including that of domains. Queries in PSQL are translated to SQL queries by a preprocessor. It supports topological relationships such as cover, not cover, overlap, etc.

A number of the query languages that have been developed are not SQL-based but ad hoc query languages. Some of these can be classified as visual spatial query languages, a natural interface for spatial data. A query language that allows users to 'draw' a query is the language Cigales [8]. In this the user must first select the type of spatial relation they are interested in and then draw a query sketch on which retrieval is based. Another approach uses an iconic query language [9]. Here the user chooses spatial relations from a predefined iconically represented set and formulates the query based on these. To keep the querying more user-friendly, only a relatively small subset of all topological relations were formulated by icons. Query-by-visual-example [10] is a visual spatial query language based on an extension of query-by-example [11]. A grid is utilized to form a template of scenes, primarily specifying cardinal directions. The grid facilitates

directional specifications, but makes it harder to specify approximate distances and topological relations independent of direction specifications. Another approach permits a sketch that is somewhat less restrictive in its retrieval matching by utilizing inference mechanisms for geometric variations in the sketch. This is the spatial-query-by-sketch [12] that is based on qualitative aspects reflecting human behavior in which the topology is of prime importance, followed by distance measures to refine the degree of match.

As a preface to discussion of spatial object querying we first describe the state of development of object query language (OQL) as specified for querying the object data management group (ODMG) object model [13]. OQL defines an orthogonal expression language, in the sense that all operators can be composed if the types of operands are valid. OQL has one basic statement for retrieval of information based on the SQL language syntax of Select-From-Where. In this it is quite similar to SQL92, with object-oriented extensions such as complex objects, object identity, path expressions, polymorphism and operator invocation.

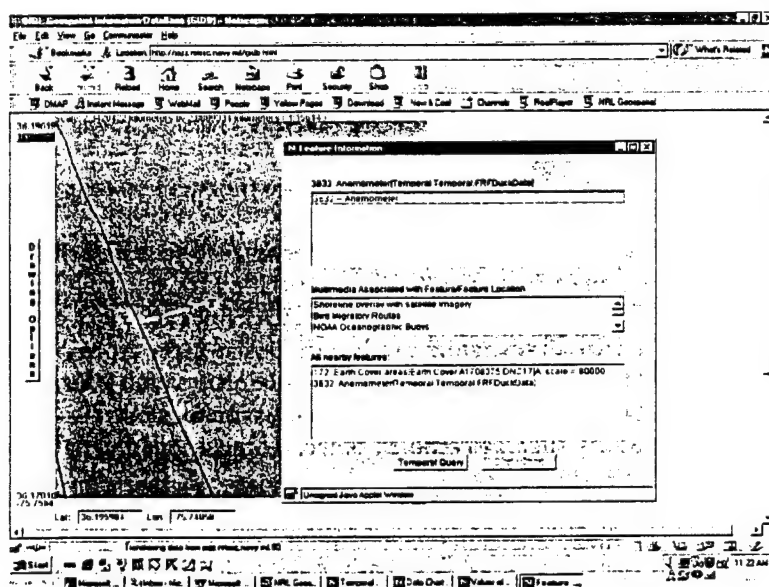
PICQUERY⁺ [14] was designed for object-oriented, multimedia knowledge-based systems. It supports fuzzy matching and temporal and object evolutionary event querying and was developed to meet needs of medical imaging requirements. The relationship terminology used by PICQUERY⁺ is somewhat different from that common in geographical domains using terms such as left of, above, and so forth. MOQL [15] provides a general query capability for multiple media and applications and it includes constructs to capture temporal and spatial relationships in multimedia data. Extensions to the WHERE clause of OQL in the form of new predicates are provided for temporal and spatial expressions and a 'contains' predicate.

3.2. Evaluation of GIDS Query Interface and Other Approaches

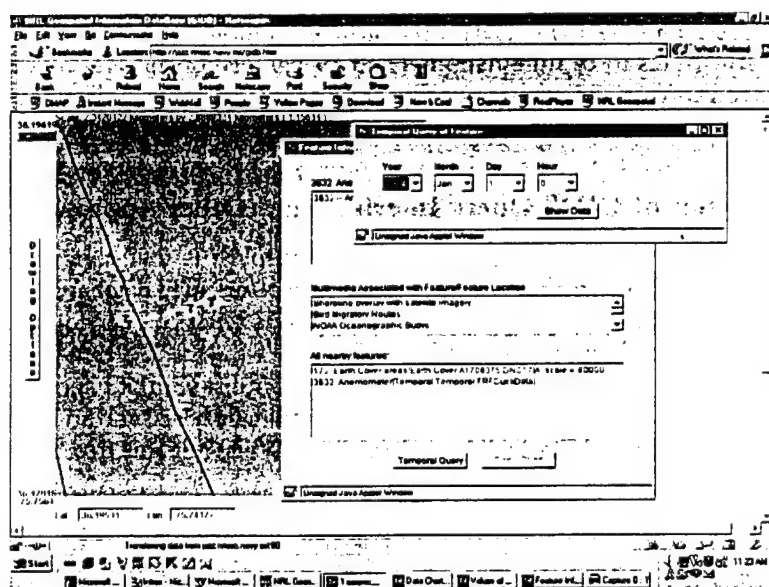
The main characteristic of our spatial querying approach is that it has a data-driven graphical interface that is dealing directly with an object-oriented database. Now we shall discuss the relationship of our approach to the various types of querying languages for spatial data that have been overviewed above.

Several aspects to SQL-based querying schemes contrast with our specific approach. A basic difference is that SQL is fundamentally relationally based and so mismatched to our totally object-oriented Smalltalk-based database. While it is true that SQL3 supports objects as ADTs which describe new types for attributes, this is in an object-relational context in which relations are still central to the SQL3 view of data [16]. Another problem with SQL is that it and its extended versions are primarily not spatially oriented. They require a user to refer to geo-referenced data via an alphanumeric language [6]. Such spatial querying requires the user to focus on language syntax and to often translate a spatial image in their mind into a non-spatial language. Thus, in this sense our data-driven graphic query interface is more comparable to the graphic query approaches described in the last section. The sequence of Figure 3(a)–(e) that follows demonstrates the benefits of the data-driven query.

First, a sensor that collects time-varying information, shown as T on the map display, is selected. Upon selecting the sensor, its available attributes along with related multimedia data and nearby features are displayed. The *Temporal Query* button is enabled to indicate that there are temporal information available with the selected anemometer.



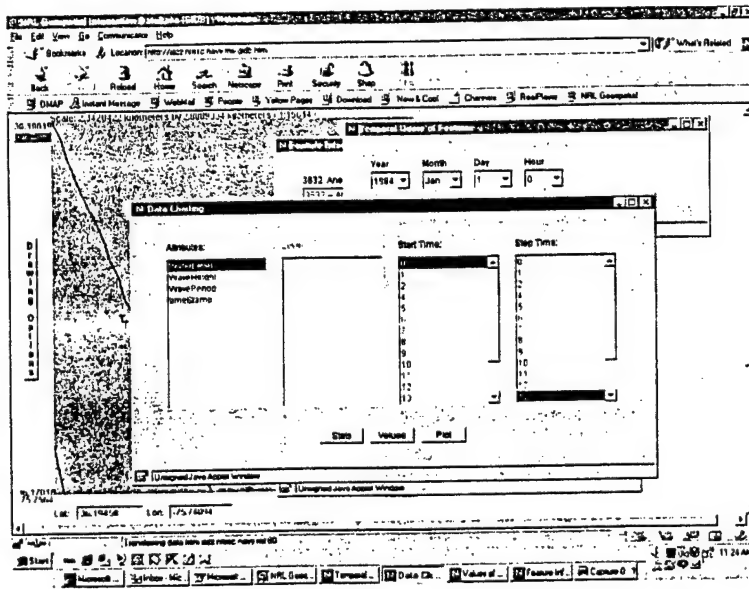
(a)



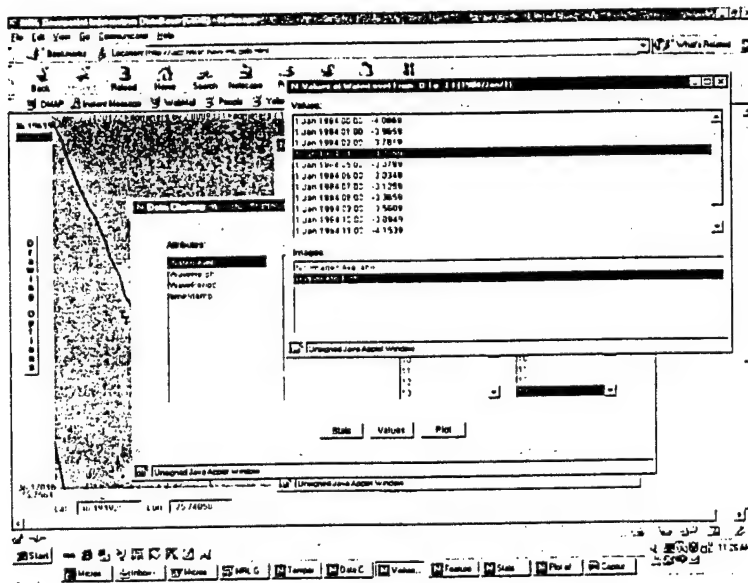
(b)

Figure 3. Data-driven query example

Upon clicking the Temporal Query button, a *Temporal Query of Feature* window is displayed. This display allows the user to select the time of interest. The year, month, day and hour selection will be based on the actual data availability for the anemometer. Selection can stop at any level, i.e. year, month, day, or hour. In this case, after selecting the hour, the *Show Data* button is selected.



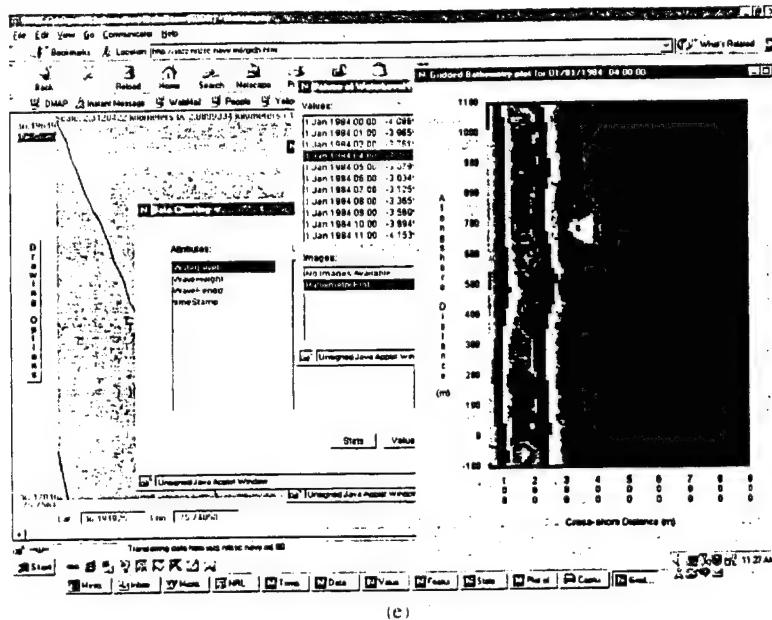
(c)



(d)

Figure 3. Continued

The anemometer during the selected time has collected four different types of information, WaterLevel, WaveHeight, WavePeriod, and timeStamp. Upon selecting WaterLevel as the information of interest, another time selection is provided. This time selection is in minutes. Upon selecting the time period of interest, any of the three options can be selected to view the data, statistics, values, and a plot.

Figure 3. *Continued*

Upon clicking *Values* to view the data, a list of all the water level information at the time period is displayed. Upon selecting one of the time instances, images that are related to this time stamp are displayed. A set of video cameras are positioned at the near locations taking video shots of the shorelines. For the selected time instance, there are no video shots available. However, there is a set of bathymetric data collected at the same time over at the same location. By selecting the Bathymetric Plot, a real time display of the bathy plot is shown.

In this example, the user did not need to have any prior knowledge of the data. The querying process was driven by the available data, which were displayed to the user at each step.

As a conclusion to our discussion on spatial query languages, we consider those query approaches based on OQL which are indeed object-oriented. Although these provide an object-oriented stance, they still differ from our approach by being structured syntactically (SQL-like) and not directly oriented toward a data-driven approach. Indeed, one of the key considerations behind the OQL design was the compatibility with the SQL [13]. It is understood that the OQL extends SQL to incorporate those unique characteristics of objects [17]. A system which has more in common with our approach is QUIVER [18]—a graphical interface to an object-oriented database, O₂. A formal user evaluation of this system indicated that it was more convenient for query formulation than the direct use of OQL. The major differences from our approach are that QUIVER is a graph-based language for general database querying and not specifically meant for spatial databases, and that it also translates queries into OQL, thus generating some overhead that our system does not incur.

4. Spatial Object Query Formalism

As in our general discussion of object-oriented queries, the first issue to be addressed is the object data semantics. Let us use the notation Σ as the universe of spatial objects. It consists of all objects that occupy some geographic location.

In defining 'geographic location', we recognize the fact that many spatial data modeling strategies rely upon approximate representations of spatial objects both for computational efficiency issues as well as the simplification of logical modeling strategies. One of the widely used approximations is the minimum bounding rectangle (MBR). Our work as well as that of many others [19–21], relies upon the use of MBRs as approximations of the geometry of spatial objects. The use of MBRs in geographic databases is widely practiced as an efficient way of locating and accessing objects in space [21]. In addition, numerous spatial data structures and indexing techniques have been developed that exploit the computationally efficient representation of spatial objects through the use of MBRs [21,4]. Another advantage to the use of an MBR representation is that all objects can be handled at the same level of dimensionality—that is, point, line and area features are all represented as 2-D objects across which operations can be uniformly applied.

Of course, the use of MBRs is inherently problematic to some degree because an MBR is an approximation of an object's true geometry. The best way to deal with this is to consider that MBRs are over-estimations of the object's extent and their impact on queries is to produce 'false drops' (objects whose MBR extent, but not its actual geometry, satisfies the query). If a query refinement is required, the underlying actual vector representation can be then used to eliminate the false drops.

The MBR is used as a representation of a feature's geometric or locational aspect. The abstract object structure we will use here is

$$O = \{O.loc, O.att\} \quad \forall O \in \Sigma$$

$O.loc$ is the component of the object that represents the method to determine the location of the bounding box. The relevant VPF non-spatial attributes for the particular type of object are represented by the component $O.att$.

In formulating a general spatial query description, we first consider the use of spatial predicates and attribute predicates. Three different spatial predicates can be given: (1) spatial predicate over an \mathcal{AOI} , (2) topological, and (3) geometric. The topological and geometric predicates allow querying of spatial relationships with other spatial data in the \mathcal{AOI} . The first aspect of our specific query environment that we must address is the \mathcal{AOI} , as it forms the context for all other discussions. The \mathcal{AOI} is the general region of concern to the user and could potentially be specified in a number of ways:

$$\mathcal{A} = \{O \in \Sigma \mid P_{area}(O.loc)\}.$$

Since there are a number of specifications for the area we represent this by the general predicate P_{area} . We want to allow for the possibility of objects that may satisfy a query but whose position (bounding box) does not entirely lie in the \mathcal{AOI} , similar to the buffer

specification used in other systems [23]. So we define

$$A^* = \{O \in \Sigma \mid O \in QR \wedge (O \in A \vee (O.loc \cap A))\}$$

where QR is the set of objects retrieved by the query.

Now that we have specified an AOI , we can formulate the two basic queries allowed—attribute queries and spatial queries:

$$\text{attribute query: } QR = \{O \in A^* \mid (O.att_k = v_k)\}$$

$$\text{spatial query: } QR = \{O \in A^* \mid P_{spatial}(O, O'), O' \in A\}.$$

3

The attribute query is interpreted as: Find all objects in the AOI such that the attributes match the corresponding values in the query. The spatial query can be phrased as: Find all the objects that have the specified spatial relation to the selected object O' . The specific forms of spatial predicates allowed will be discussed shortly.

We have described the form of individual queries so far, but it is also necessary to allow a sequence of nested queries, Q_1, Q_2, \dots, Q_i . This is possible since the result of a query Q_i is a set of objects

$$QR_i \text{ where } QR_i \subseteq A \text{ (or } A^*).$$

The three object classes use the same spatial representation (bounding boxes) and so the query operations are closed. Thus, we have for nested queries

$$\text{attribute query: } QR = \{O \in QR_i \mid (O.att_k = v_k)\}$$

$$\text{spatial query: } QR = \{O \in A^* \mid P_{spatial}(O, O'), O' \in QR_i\}.$$

This simply means that the query can refer to the set of objects obtained by a previous query, i.e. the set of objects retrieved can be based on the result QR_i of a previous query Q_i .

In order to reflect the possibility of exactly which objects are returned by a spatial query when allowing sets of objects to be related, we formulate the following query result:

$$QR = \{O \vee O' \mid P_{spatial}(O, O'), O \in QR_i, O' \in QR_j\}.$$

The above query formulation selects the returned result as only the objects O or O' . As a result, some information is lost regarding the specific objects in the other set to which the resulting objects are related. QP is defined as an extension to QR that returns both the objects O and O' :

$$QP = \{(O, O') \mid P_{spatial}(O, O'), O \in QR_i, O' \in QR_j\}.$$

QP is of course a different form of result set (object pairs) than the QR s that have only objects. That is

$$QP \subseteq QR_i \times QR_j.$$

Since QP is a set of pairs of objects, the simple queries above cannot be nested with results from QP . That is, we do not maintain closure of query nesting if a QP result were used.

The types of predicates we will consider are geometric and topological predicates. The geometric predicates use either distance or positional functions:

$$P_{\text{geometric}}(O, O') = (G(O.\text{loc}, O'.\text{loc}) (<, =, >) .N)$$

where G is a distance function and N is some distance value or just

$$P_{\text{geometric}}(O, O') = (G(O.\text{loc}, O'.\text{loc}))$$

where G is a positional function which is Boolean.

For distance functions based on the bounding box, the distance can be based either on a centroid-to-centroid or boundary-to-boundary measure. For the latter, the semantics of the query might require either nearest or farthest boundary measures.

The topological predicate describes relationships among neighborhoods. Egenhofer [2] defines nine relationships that completely describe topological relationships among spatial entities. For the spatial data, a strong assumption is made that an ROI has been selected to localize a search region. A boolean value is used to determine spatial relationships among spatial entities. All spatial entities that are evaluated *true* with respect to the topological relationship are added as a query result. The topological predicates we can specify are again based on $O.\text{loc}$ as formulated from the bounding box:

$$P_{\text{topological}}(O, O') = (T(O.\text{loc}, O'.\text{loc}))$$

where T is a Boolean function representing typical topological functions that can be used, such as totally contains, intersecting, and non-intersecting.

5. Implementation of Spatial Query Framework

The GIDS serves as an object-oriented database server that allows objects to be accessed via a Java applet. More detailed discussion on the GIDS can be found in Chung [24]. GIDS hosts several different data types: vector, raster, text, multi-media, industry standard images, and temporal.

Raster, text, and a large portion of the vector data are based on the National Imagery and Mapping Agency (NIMA) standards. A triad of mapping data, namely vector product format (VPF) [25], raster product format (RPF) [26], and text product standard (TPS) [27] are disseminated by NIMA to military users. Each format is produced to serve a specific purpose to military users. Vector data are used for analysis and for providing more control over display, i.e. display can be decluttered compared to raster

display. On the other hand, the raster data are used primarily for situational orientation. Text products provide textual descriptions concerning hydrographic and aeronautical information.

VPF is designed to support large geographic databases using flat files or a relational structure. VPF specifies a geo-relational framework based on a vector data model. This format has the most complex specification among the triad. Most VPF products are digitized from scanned-in paper maps, air photos, satellite data, etc. The uses of VPF include: (1) distribution of cartographic data products, (2) direct query capability, and (3) data quality monitoring and recording. Three basic types of information specify VPF data: non-spatial properties (attributes), geometric properties (coordinates), and topological properties (connectivity and contiguity relationships). A spatial entity in VPF is at a feature level. Four types of features are used: point, line, area, and text. An overview of VPF and topology is discussed in Chung [28]. A discussion of conversion from flat file structure to object-oriented format for VPF is provided in Arctur [29].

RPF is a standard database structure for arrays of pixel values. Both compressed and uncompressed forms of raster data are available. RPF data is generated from scanned charts as well as SPOT imagery. An RPF image is defined by frames. Each frame is sub-divided into subframes and each frame and subframe represents a specific geographic region covering an area specified by four corners of a rectangular boundary. Every subframe has a pixel array that provides color index, value or intensity for each corresponding pixel location. Figure 4 shows an example of the display of both VPF and RPF data types for the Persian Gulf region. Vector plots of shorelines, islands, rivers and currents are also shown in this example. We emphasize here that data are not integrated merely at a visual level; all data displayed, regardless of format, can be queried regarding attributes, metadata, etc.

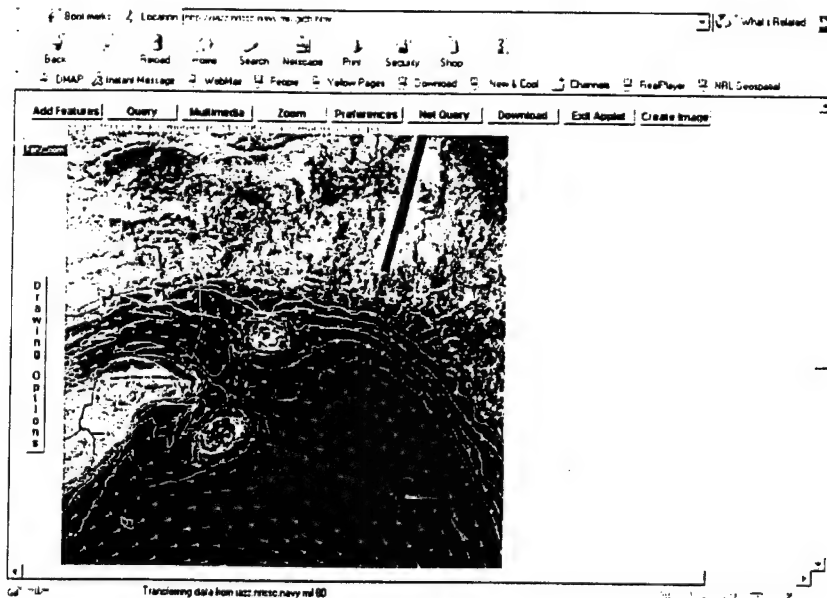


Figure 4. Integrated VPF and RPF data

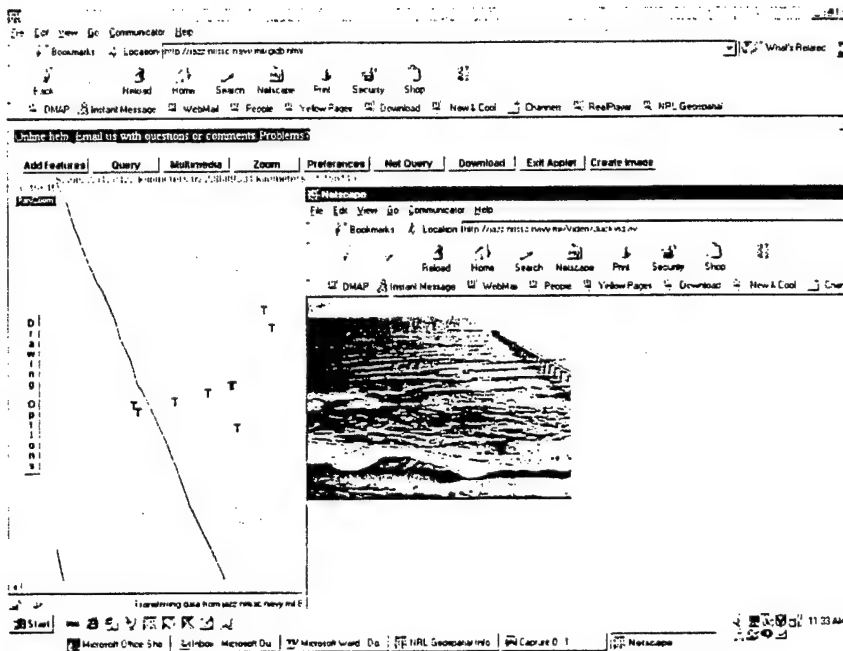


Figure 5. Integration of temporal data and avi files

TPS is the third NIMA format that provides digital textual description of hardcopy publications published by NIMA, e.g., *American Practical Navigator*. The format of TPS is paragraphs with associated figures and tables using standard generalized markup language (SGML). The content of TPS is used as a secondary information source to augment paper or digital charts. Every TPS product has an indexed gazetteer that provides relationships among name, geo-location and paragraph location within the document. The indexed gazetteer is the main link that is used in generating geo-referenced TPS object data.

The GIDS can currently host video files (mpg, avi, and mov), audio files (wav), and industry standard image (jpg and gif) formats. Each of these multi-media data sources is referenced to a certain spatial region. For example, GIDS has an audio clip of Hurricane Fran that hit North Carolina in 1996 that is referenced to the spatial location of North Carolina. Figure 5 shows the use of 'T' symbols on a vector map that represent the availability of temporally registered data for particular geographic locations. Clicking on one of the symbols brings up a web browser in which the available text is displayed. This figure also shows an avi file of the currents for the area. Figure 6 shows a variety of multi-media data, including coastal surveys and space shuttle imagery.

Temporal data persisted in GIDS are collected by the Field Research Facility in Duck, North Carolina. The temporal data spans over almost two decades, from 1980 to 1999. Each time-varying information set was collected by sensors that record changing waves, winds, tides, and currents [www.frf.usace.army.mil]. These temporal data are referenced spatially by the sensor's location. Also, each sensor has a reference to the time-varying information it records.

All the data that GIDS currently has persisted have spatial reference. Each of the data sets has a spatial access mechanism embedded in its class hierarchy. Two factors

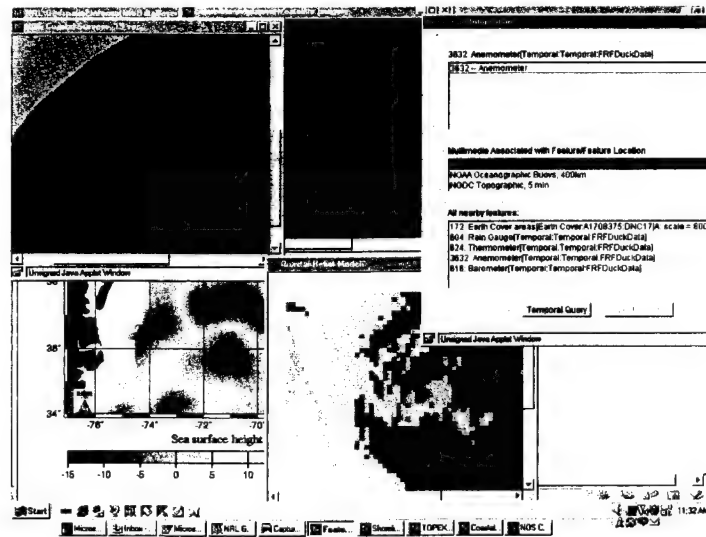


Figure 6. Multi-media display

determined the design of the spatial access mechanism in GIDS: balancing the tree, and ease of access. In other words, for the multi-media data sets, a global spatial access mechanism is used to spatially index the multi-media data. Due to the complexity in the VPF data organization and the volume of data, a spatial access mechanism is embedded within each thematic layer [30].

In dealing with multiple data formats and data types, one of the primary issues deals with the integration. In GIDS, spatial data integration is achieved by the use of a spatial access mechanism as shown in Figure 7. For each data type, its data structure is implemented according to the specification. However, each data type class hierarchy always contains the spatial access mechanism as one of its instance variables to index spatial objects.

5.1. Spatial Query Over AOI

An *AOI* specification can be created in one of three ways (refer back to Figure 2 for illustration): (1) specify origin and corner location (a diagonal of a rectangle), (2) specify a center location and a radius, and (3) specify a location with horizontal and vertical distance from that location. Using one of the three methods of specifying an *AOI*, the spatial query is evaluated by using a quadtree. The following method is executed to perform the spatial query:

```
quad returnSetOfIntersectingFeatures: aRect.
```

A **quad** is an instance of Spatial Data Manager that manages the spatial index, namely a quadtree, and **aRect** is the *AOI* specified by a user. The use of polymorphism allows

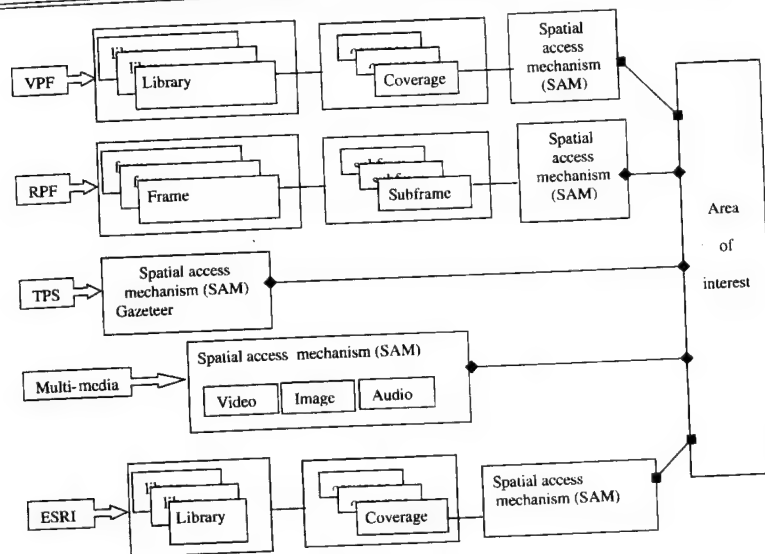


Figure 7. Spatial data integration using spatial access mechanism

each class of data to respond to a **quad** query, thus allowing us to deal uniformly with heterogeneous data types. A quadtree is implemented as the sam for each data set in GIDS due to its simple and intuitive design [3].

The method **returnSetOfIntersectingFeatures: aRect** uses an **intersect** method to determine if a bounding box of a spatial data set and a cell intersect. All cells that intersect the bounding box of a spatial data set are candidates for the requested spatial query. A path of all the intersecting cells should be along one branch of a quadtree. Since data is stored in a *features* collection, the spatial query returns all elements in the *features* collection of each intersecting cell. Remember that only one quadtree is used for all three data types (VPF, RPF and TPS). The *features* collection maintains all VPF, RPF, and TPS data. All spatial query predicates can be applied to each data type independent of the dimension, shape, and semantics.

A spatial search always begins at the root of a tree. This search strategy is not optimal nor efficient considering the general access in a geographic information system (GIS) environment. That is, users typically start from an area and 'zoom in' and 'zoom out' needing to access only one branch or sub-branch of a quadtree. Search and retrieval performance can therefore be improved by leveraging this access paradigm. Cobb [31] provides a discussion on approaches to improve performance by using a splay tree. Alternatively, a 'working root' can be defined as the cell that contains spatial data in *features* and is a parent to all other cells that intersect the *AOI*.

The query model is not limited to a static set of *a priori* identified datasets. For example, Figure 8 shows a list of available data sources (the **Add Feature: window**) in

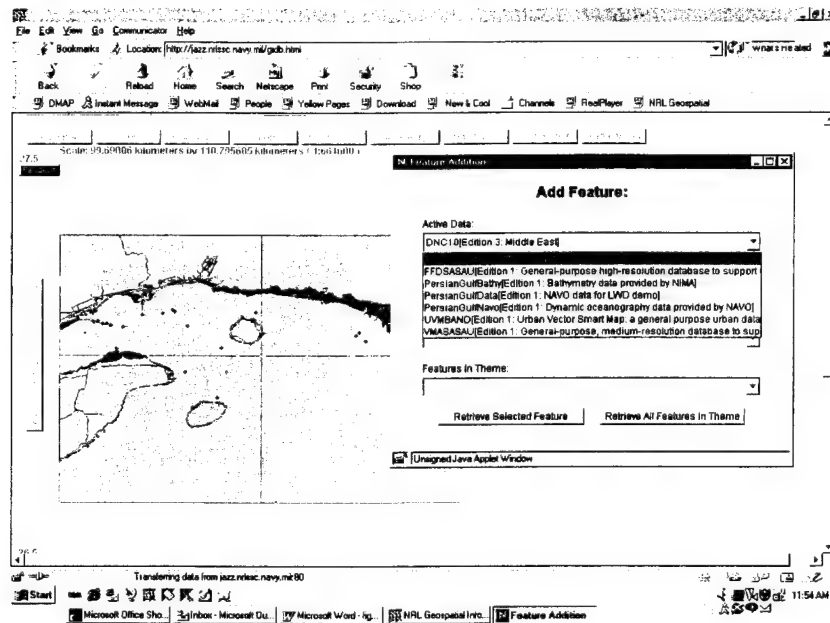


Figure 8. Integrating new data sources for query evaluation

different formats that can be integrated into previous query results displayed on the map. A completely new data source can be selected, or this option can be used to add new types of features from a previously used source. The integration of the data types over *AOI* through the quadtree is the concept that makes this feasible.

5.2. Topological Spatial Query Predicate

Currently, only the *intersect* relationship among objects is implemented. Two object types are assumed to determine the intersection relationship. Two-stage intersection computation is executed: MBR, then object geometry. An approximate *intersect* relationship is first imposed on the MBRs by the virtue of a spatial query over an *AOI* using a quadtree. As a retrieval from a quadtree is in process, each object that is within the *AOI* must be tested to determine if the object is one of the object classes specified as a part of the query criteria. Only those objects that are of the object classes are retrieved.

An MBR intersection does not guarantee actual object intersection by the pure definition of MBR. Therefore, the geometry of each object is used to determine actual object-to-object intersection. Each object group can be of different object types (i.e. among RPF, TPS, and VPF) or a result of any previously executed query result.

5.3. Geometric Spatial Query Predicate

Two geometric relationships regarding the distance metric are implemented: *within* and *greater than*. These relationships are mutually exclusive. The upper-bound distance for *greater than* computation is the *AOI* extent. Figure 9 shows an example of the geometric query, 'find all transportation lines within 0.5 m from river lines', over the Persian Gulf region. The query, which is partially hidden in the figure, is stated completely as, 'Q3-Q1 within distance 0.5 m from Q2', where Q1 is all transportation lines and Q2 is all river lines.

5.4. Attribute Query

Once objects that meet the specified query criteria over the *AOI* have been retrieved, each object's attribute can be readily accessed based on the data encapsulation property of object-oriented technology. Two steps are involved in evaluating any query that specifies attribute constraints. First, all object types within the *AOI* that are specified as query criteria must be retrieved from the quadtree. Secondly, each retrieved object must be evaluated to determine if the specified attribute condition is met or not.

Remember that only VPF data have attributes. A VPF object at an abstract class *VPFFeature* maintains attribute information as part of its state information. Therefore, we can inspect the attribute information for all previously retrieved VPF features.

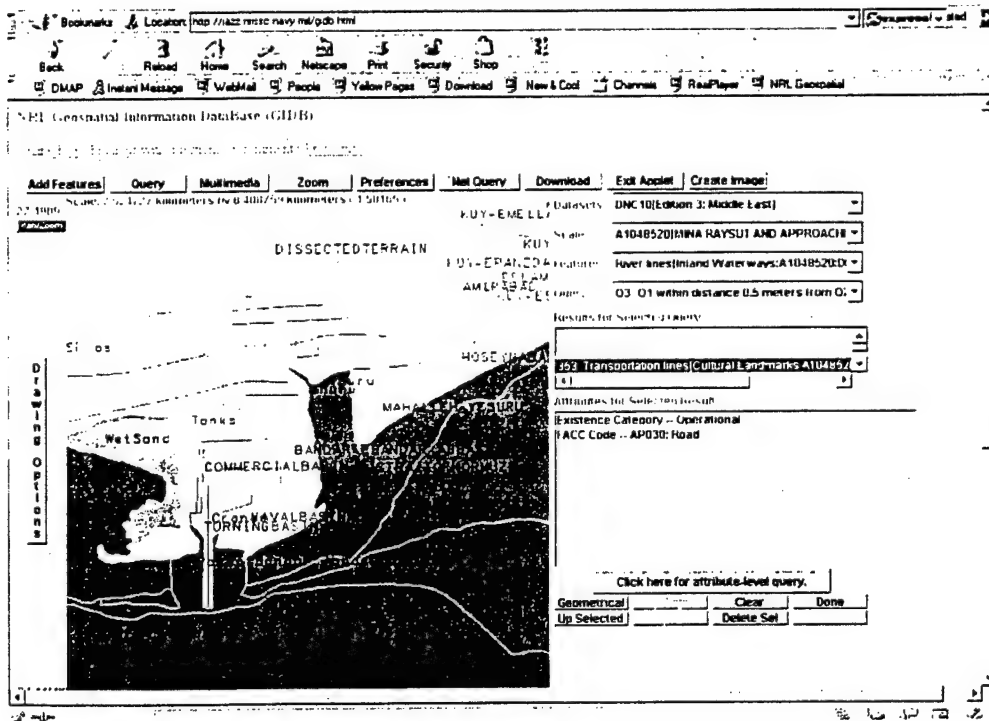


Figure 9. A geometrical query of transportation lines within 0.5 m from river lines is displayed

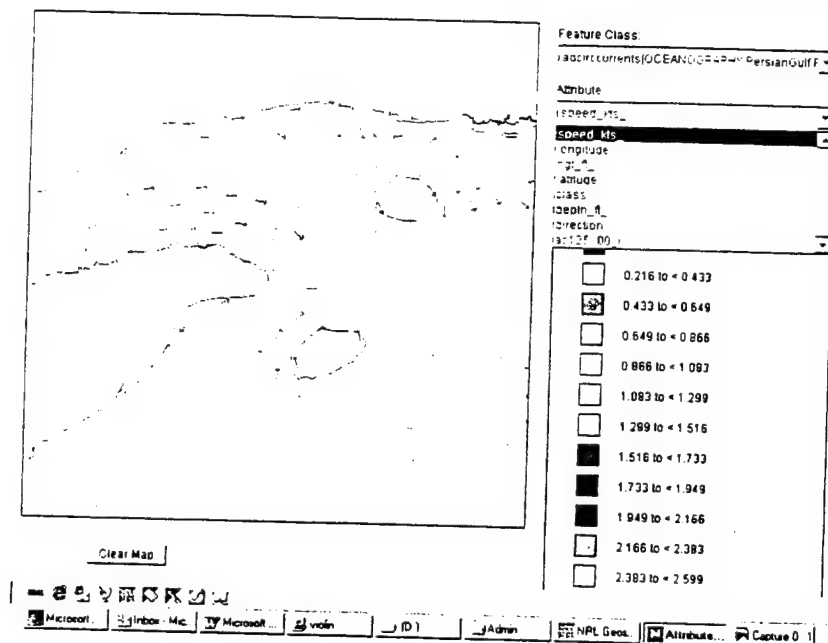


Figure 10. Representation of attribute query results

Query processing at the attribute level implies querying at a feature level. Each feature has attributes and values associated with each attribute for that feature. Thus, attribute query is an analysis at a feature level. A sample is *'Find all bridges within the specified AOI that can be used as a pedestrian walkway'*. A pedestrian walkway is a value (17) of an attribute *transportation usage category (tuc)*. Another example of an attribute query is illustrated in Figure 10. In this case, circulation currents are shown in a vector, or 'hedgehog' plot, while ranges of values in knots are differentiated by a colorscale. This example also demonstrates the significance of the visual representation of query results related to spatially registered data.

5.5. Nested Query

The query model developed has the capability to allow the construction of complex queries from simple queries. It is believed that users ask simple questions, then use these simple questions as building blocks to create complex queries. Therefore, it is important to be able to maintain a working set of previous queries.

In order to manage and maintain each query and its results, query indexing is necessary. In other words, each query and its results must be indexed to be readily accessible. To allow query nesting, the following minimum information regarding each query must be maintained: query criteria, a query result of object type 1, and a query result of object type 2. For relative queries such as geometrical or topological, at least two object types are required, e.g. transportation lines (object type 1) that are within 20 miles of river lines (object type 2). This information is maintained in a list. Therefore, each query is implicitly indexed by its position in the list. With this information, any user

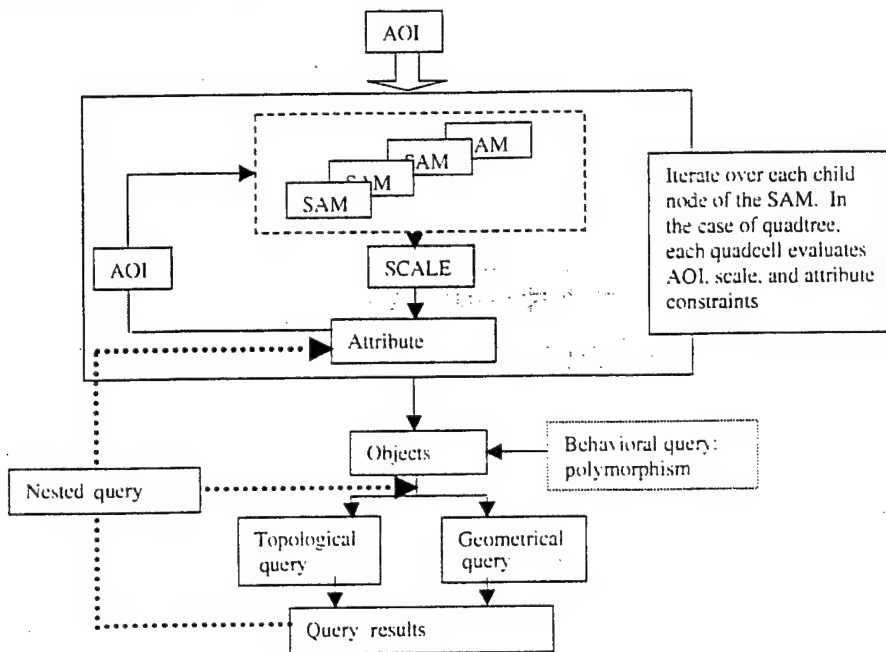


Figure 11. Spatial query processing in GIDS

can readily formulate a complete query criteria when making a nested query based on previous queries. Secondly, the query results do not have to be recomputed, which provides overall performance improvement. As spatial objects in the *AOI* are searched, a parallel query on each object continues relative to the scale and attribute constraints. Those spatial objects that meet all the constraints are retrieved. Once the objects are retrieved, further query continues at an object level. In other words, through polymorphism, each object is capable of responding to certain requests. Based on the implementation of the object behavior, each object would respond appropriately to requests such as display. An implementation of the spatial query model is summarized in Figure 11.

Our spatial query processing begins by specifying an *AOI*. This *AOI* is specified as an MBR. The specified MBR is sent to all the quad trees of all the different data types. All the objects that meet the spatial constraint are then evaluated against the scale specified by the user, followed by the attribute constraints. Once all the objects that meet the *AOI*, *scale* and *attributes* are retrieved, further query such as topological or geometrical queries are processed, if desired. Any query beyond the initial *AOI*, *scale* and *attribute* query is considered as part of a nested query.

Thus, from the spatial data domain and the design of the GIDS, spatial query processing is achieved through a parallel evaluation of the *AOI*, *scale*, and *attribute* constraints as the *spatial* search continues on a *sam*. Once the spatial search finishes over the *sam*, relative and behavioral (polymorphic) queries can be continued on those spatial objects that met *AOI*, *scale*, and initial *attribute* constraints. Nested queries use the query results as arguments for the relative comparison and computation. Further attribute constraints can be imposed on those spatial objects that met previous query requirements.

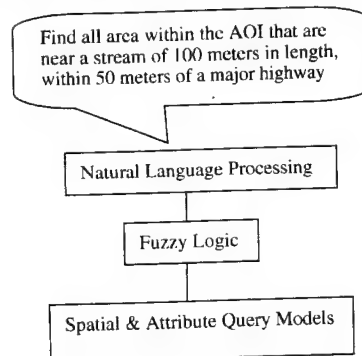


Figure 12. Extensions for imprecise querying

6. Conclusion and Future Work

An object-oriented approach is recognized as one of the most promising techniques for modeling complex data such as spatial data. In querying multiple heterogeneous data sources, the inherent properties of an object-oriented approach, i.e. data encapsulation and polymorphism, allow for flexibility in achieving a data-driven approach of query processing. A formal model of spatial querying requires three parameters: *AOI*, *scale*, and *attributes*. An initial search is conducted over the spatial region. As the spatial search continues, a parallel evaluation of the *scale* and *attributes* is performed. Due to the polymorphism and data encapsulation properties of the object-oriented approach, relationship (topological and geometrical) and behavioral queries can be further evaluated.

We are examining approaches to extensions of predicates to allow linguistic expressions such as
 '... about 5 kilometers ...'
 '... slightly overlapping ...'

The approach we have been considering for the semantics of such applications involves the use of fuzzy logic [31]. Figure 12 shows the extensions and future work that are anticipated.

References

1. M. Egenhofer (1990) Interaction with geographic information system via spatial queries. *Journal of Visual Languages and Computing* 1, 389-413.
2. M. Egenhofer (1989) Spatial query languages. *Ph.D. Thesis*, University of Maine, (unpublished).
3. V. Gaede & O. Gunther (1998) Multidimensional access methods. *ACM Computing Surveys* 30, 170-231.
4. H. Samet (1989) *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley, Reading, MA.

5. M. Cobb, M. Chung, H. Foley, F. Petry, K. Shaw & V. Miller (1998) A rule-based approach for the conflation of attributed vector data. *GeoInformatica* **2**, 7–35.
6. M. Egenhofer (1994) Spatial SQL: a query and presentation language. *IEEE Transactions on Knowledge and Data Engineering* **6**, 86–95.
7. N. Roussopoulos & C. Faloutsos (1988) An efficient pictorial database system for PSQL. *IEEE Transactions on Software Engineering* **14**, 639–650.
8. D. Calcinelli & M. Mainguenaud (1994) Cigales, a visual query language for a geographical information system: the user interface. *Journal of Visual Languages and Computing* **5**, 113–132.
9. Y. Lee & F. Chin (1995) An iconic query language for topological relationships in GIS. *International Journal of Geographical Information Systems* **9**, 25–46.
10. D. Papadias & T. Sellis (1995) A pictorial query-by-example language. *Journal of Visual Languages and Computing* **6**, 53–72.
11. R. Elmasri & S. Navathe (2000) *Fundamentals of Database Systems*, 3rd. edn. Addison-Wesley, Reading MA, pp. 310–318.
12. M. Egenhofer (1997) Query processing in spatial-query-by-sketch. *Journal of Visual Languages and Computing* **8**, 403–424.
13. R. Cattell et al. (eds) (2000) *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann, San Francisco, CA.
14. A. Cardenas, I. Jeong, R. Taira, R. Barker & C. Breant (1993) The knowledge-based object-oriented PICQUERY⁺ language. *IEEE Transactions on Knowledge and Data Engineering* **5**, 644–657.
15. J. Li, T. Ozsu, D. Szafron & V. Oria (1997) MOQL: a multimedia object query language. *Proceedings of the International Workshop on Multimedia Information Systems*, Como, Italy, pp. 19–28.
16. J. Ullman & J. Widom (1997) *A First Course in Database Systems*, Prentice-Hall, New Jersey. (Chap 8.)
17. S. Cluet (1998) Designing OQL: allowing objects to be queried. *Information Systems* **23**, 279–305.
18. M. Chavada & P. Wood (1997) Towards an OMDG-compliant visual object query language. *Proceedings of the 23rd VLDB Conference*, Athens, pp. 254–264.
19. M. Nabil, J. Shepherd & A. H. H. Ngu (1995) 2D projection interval relationships: a symbolic representation of spatial relationships. In: *Advances in Spatial Databases: 4th Symposium, SSD '95*, Maine, U.S.A., pp. 292–309.
20. J. Sharma & D. M. Flewelling (1995) Inferences from combined knowledge about topology and direction. In: *Advances in Spatial Databases: 4th Symposium, SSD '95*, Maine, U.S.A., pp. 279–291.
21. E. Clementini, J. Sharma & M. Egenhofer (1994) Modelling topological and spatial relations: strategies for query processing. *Computers and Graphics* **18**, 815–822.
22. H. P. Kriegel, M. Schiewietz, R. Schneider & B. Seeger (1989) *Performance comparison of Point and Spatial Access Methods In Design and Implementation of Large Spatial Databases* (A. Buchmann, O. Gunther, T. Smith & Y. Wang, eds), Springer-Verlag, Santa Barbara, CA, pp. 89–114.
23. V. Gaede & W.-F. Rieker (1994) Query evaluation in the object-oriented GIS GODOT. In: *Proceedings of the 1st AGDM Workshop*, Delft, Holland.
24. M. Chung, R. Wilson, R. Ladner, T. Lovitt, M. Cobb, M. Abdelguerfi & K. Shaw (2001) The geospatial information distribution system (GIDS). In: *Succeeding with Object Databases*, Chapter 17 (A. Chaudri & R. Zicari, eds), John Wiley & Sons Inc., New York, pp. 357–378.
25. National Imagery and Mapping Agency (1993) Military Standard: Vector Product Format. Draft Document No. MIL-STD-2407.
26. U.S. Department of Defense (1994) Military Standard Raster Product Format. MIL-STD-2411.
27. National Imagery And Mapping Agency (1995) Text Product Format Standard Draft.
28. M. Chung, M. Cobb, K. Shaw & D. Arctur (1996) An object-oriented approach for handling topology in VPF products. In: *Proceedings GIS/LIS 1995, Vol. 1*, Nashville, TN, pp. 182–192.
29. D. Arctur, M. Chung, M. Cobb & K. Shaw (1995) Comparison and benchmarks for import of vector product format (VPF) geographic data from object-oriented and relational database files. In: *Proceedings of Fourth Symposium on Spatial Databases, SSD 95*. Springer-Verlag, Berlin, pp. 368–384.

GEOINFORMATICA

An International Journal of
Geomatics Science

EDITOR-IN-CHIEF: PATRICK BERGOUNNOUX

Regional Co-Editors

- ♦ Africa/Europe : Andrew Frank
- ♦ America : Max Egenhofer
- ♦ Asia/Australia : George Benwell

VOLUME 2, NUMBER 1, MARCH 1998

A Word from the Editor-in-Chief <i>Patrick Bergounoux</i>	5
A Rule-based Approach for the Conflation of Attributed Vector Data <i>Maria A. Cobb, Miyi J. Chung, Harold Foley III, Frederick E. Peiry, Kevin B. Shaw, and H. Vincent Miller</i>	7
Sorting Spatial Data for Sampling and Other Geographic Applications <i>Alan Scafield</i>	37
Managing, Modeling, and Visualizing High-dimensional Spatio-temporal Data in an Integrated System <i>Lars Bernard, Benno Schmidt, Ulrich Streit, and Christoph Uhlenkücken</i>	59
3D-GIS for Urban Purposes <i>Alexander Köninger and Sigrid Bartel</i>	79

A Rule-based Approach for the Conflation of Attributed Vector Data

MARIA A. COBB,¹ MIYI J. CHUNG, HAROLD FOLEY III, FREDERICK E. PETRY, KEVIN B. SHAW
Naval Research Laboratory, Stennis Space Center, MS 39529
email: cobb, chung, foley, petry, shaw@nrlssc.navy.mil

II. VINCENT MILLER²
Planning Systems, Inc., Stennis Space Center, MS 39529

Received May 31, 1996; Revised August 13, 1997; Accepted November 11, 1997

Abstract

In this paper we present a complete approach for the conflation of attributed vector digital mapping data such as the Vector Product Format (VPF) datasets produced and disseminated by the National Imagery and Mapping Agency (NIMA). While other work in the field of conflation has traditionally used statistical techniques based on proximity of features, the approach presented here utilizes all information associated with data, including attribute information such as feature codes from a standardized set, associated data quality information of varying levels, and topology, as well as more traditional measures of geometry and proximity.

In particular, we address the issues associated with the problem of matching features and maintaining accuracy requirements. A hierarchical rule-based approach augmented with capabilities for reasoning under uncertainty is presented for feature matching as well as for the determination of attribute sets and values for the resulting merged features. Additionally, an in-depth analysis of horizontal accuracy considerations with respect to point features is given.

An implementation of the attribute and geometrical matching phases within the scope of an expert system has proven the efficacy of the approach and is discussed within the context of the VPF data.

Keywords: conflation, deconflation, rubber-sheeting, digital mapping, VPF, object-oriented

1. Introduction

Conflation is typically regarded as the combination of information from two digital maps to produce a third map which is better than either of its component sources. The history of map conflation goes back to the early to mid-1980s. The first clear development and application of an automated conflation process occurred during a joint United States Geological Survey (USGS)-Bureau of the Census project designed to consolidate the agencies' respective digital map files of U.S. metropolitan areas [20]. The enormity of the task warranted an automated computerized system, the realization of which provided an

¹Current affiliation is the University of Southern Mississippi, Department of Computer Science and Statistics, Hattiesburg, MS 39406-5106.

²Current affiliation is Energy Spatial Analysis Research Laboratory, 1430 Tulane Avenue, New Orleans, LA 70112.

tion for much of the theory and many of the techniques used today. Since
s, including commercial GIS vendors, have implemented conflation tools
lications.

onflation of maps is a complex process that must utilize work from a wide
cts, including pattern recognition, statistical and graph theory, and a
euralistics that hardcopy cartographers have used for decades to enhance
increase information content of paper maps.

f maps typically is needed because: (1) users wish to update their mapping
thout losing legacy data which may not be included in the new information;
ource may be more accurate with respect to, e.g., positional or attribute
r (3) one map source contains information missing in another, such as
ures, feature attributes or even entire coverages. Our motivation for this
all three of these reasons and represents an effort to more fully utilize
spatial data as well as spatial properties in an "intelligent" type of system
ly mirrors the way that human experts would manually conflate mapping

ing section provides background information on the steps involved in
ion 3 introduces the issues associated with map conflation and presents our
he problem; the section deals specifically with the issues of feature
confliction (removing conflicts between matching features), and positional
ements, and includes our complete conceptual design for conflation.
nts our implementation work on feature matching, and Section 5 follows
ary and indications of future work.

id

i multi-step, iterative process that involves positional re-alignment of
ips, identification of matching features and positional and attribute
i positively identified feature matches. This section contains background
the traditional ways in which these processes have been performed.

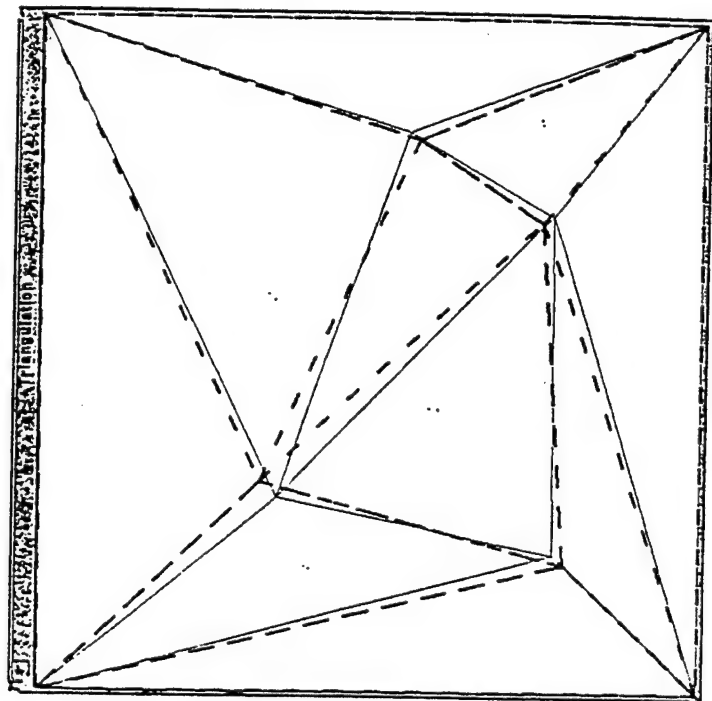
sheeting

n map conflation involves a process commonly known as *rubber-sheeting*,
linates of two maps are iteratively brought into alignment, or *registered* to
nally, a rubber-sheet transformation is a mathematical function from one
ion to another, preserving topology in the process [13]. This technique
ie from the logical analogy of that of stretching a piece of rubber to fit over
ery often (e.g., [8], [17]), one of the source maps is taken to be the base map
econd map is aligned.

ing techniques typically subdivide the map areas into triangular-shaped
ulation is a term that can refer to either the actual triangular regions or the

method that is used to generate the regions. One such triangulation method is the *Delaunay triangulation*, considered by some to be the "best" triangulation for rubber-sheeting. Gillman [13] provides a discussion of well-defined triangulations, of which the Delaunay triangulation is one, along with properties of the Delaunay triangulation that make it especially suitable for rubber-sheeting techniques. Two characteristics distinguish the Delaunay triangulation: (1) no vertex other than the three forming a triangle are contained within the circumcircle of that triangle, and (2) the minimum angles of the triangles formed are maximized [16]. These characteristics imply that fewer long, thin triangles are formed. This is desirable because the absence of such triangles leads to better results in rubber-sheeting.

Additionally, the Delaunay triangulation is both locally well-defined and therefore, also globally well-defined. For locally well-defined triangulations, it can be determined whether any three points form a triangle independently from the remaining set of points. This property implies the condition of globally well-defined triangulations in which the formation of the triangles is not dependent upon the order of processing of the points. In conflation, it is crucial that a well-defined triangulation be used in order to guarantee a unique triangulation of the input for defining the rubber-sheet transformations. Figure 1 shows an example of rubber-sheeting using Delaunay triangulation. In this figure, solid lines represent the triangulation of the base map, while dashed lines represent the



of the rubber-sheeted map. Lighter points are the original rubber-sheet darker points are the new stretched points.

ive for conflation is to form identical triangulations over both the base and map. Triangulation is first performed over the base map on a selected set of for each triangle in the base map, a corresponding triangle is created in the map. Transformation coefficients that determine scaling, translation and on-matched points within a single triangle are derived upon the formation of These coefficients are used to reposition features during rubber-sheeting.

Feature matching

to create corresponding triangles in the rubber-sheet map, we must be able to within a certain degree of confidence, points which are identical between the s known as *feature matching*. Feature matching, as the name indicates, identification of features from different maps as being representations of the same entity. This is illustrated in figure 2 with a snapshot from our application pictured are coverages from two libraries within a VPF product showing two which appear to be separate representations of the same railroad. Attribute available in the attribute editor windows along the sides indicates that identical, the information is not contradictory either. For example, while any for one is given as "main line," railroad category for the other is simply

feature matching results are imperative for rubber-sheeting. Match criteria for system must be explicitly defined, and can include properties related to lines, geometry, topology, graph properties, neighborhood groupings and arity. Rosen and Saalfeld [19] describe a feature matching process in which are first matched, followed by the matching of lines (1-cells) and areas

.) presents an iterative process of conflation in which feature matching is

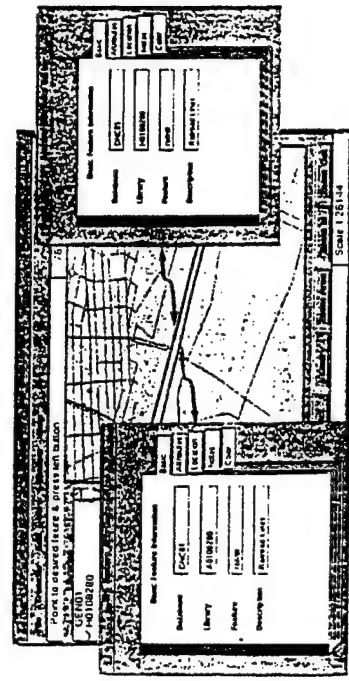


Figure 2. Example of feature matching problem.

performed first according to the strongest set of criteria and progressing to the weakest. After matches for each iteration are identified, the maps are aligned via rubber-sheeting techniques. The matching process is then repeated. When this process for a given set of criteria finds no matches, the next weaker set of criteria is used, etc. The conclusion of Saalfeld's discussion is that proximity is a necessary, although insufficient, condition for feature matching. Local configuration measurements, such as the spider function [19] and the degree of a 0-cell, are also used to increase the probability of obtaining correct matches.

Rosen and Saalfeld [19] present a taxonomy of feature matching criteria and include the concepts of dependent and independent criteria, as well as topological, geometrical and configuration-based tests. While this work is extremely important in identifying match criteria for the Bureau of the Census project, our problem differs from theirs in at least two ways. First, the maps used at the Census Bureau for conflation were very similar in content and scale, consisting primarily of road networks of metropolitan areas. In contrast, we are concerned with the more general ability to conflate map coverages of widely varying scales and applications. Also, the number of repeated features could vary greatly depending upon the similarity of the map coverages being merged. For example, the combination of earth cover with vegetation coverages would most likely have many repeated features, while the combination of earth cover with cultural coverages would have very few. Second, little or no attribution was available for use in the conflation process for the Census Bureau project. However, there now exist vector data which are highly attributed, and we believe that the incorporation of the attribute data into the conflation process can greatly enhance the quality of the resulting product.

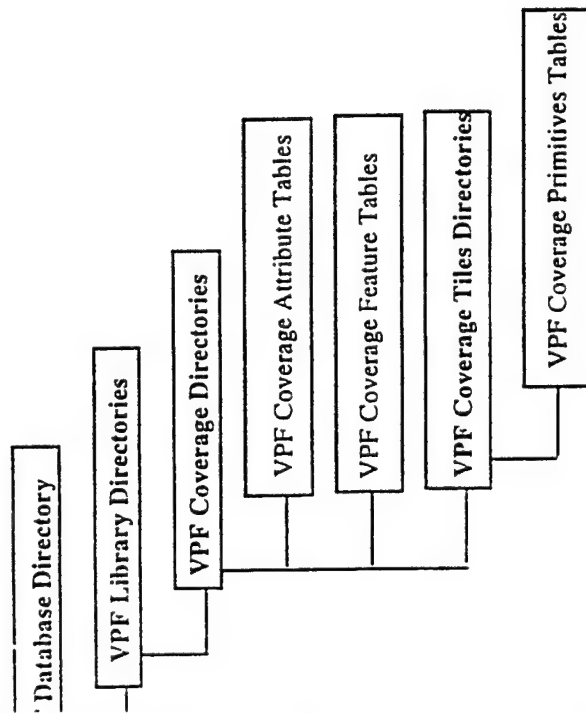
3. Approach

While our approach to conflation is in general applicable to any attributed mapping data, for implementation purposes we have concentrated on VPF data, due to its rich information content and widely varying application. In order to more fully illustrate our approach, we first present an introduction to the relevant properties of VPF.

3.1. Vector Product Format (VPF)

VPF provides a standard relational format and organization techniques for large geographic databases based on a geo-relational vector data model. VPF allows for the representation of collections of different geographic entities. These are expressed as *products* that contain source data from maps, air photos, satellite data, etc. A product specification is derived from individual conceptual designs together with the VPF standard. Some of the possible uses of VPF include: the distribution of GIS or cartographic data products, direct querying, and data quality monitoring and recording.

3.1.1. Data model. The foundation of the VPF data model is a hierarchical directory/file structure arranged in the manner shown in figure 3. A VPF *database* is a directory that



Directory Structure.

VPF subdirectories. Each library has a predefined geographic extent and each library directory has *coverage* subdirectories. All of the data within a directory is organized hierarchically (e.g., transportation, vegetation) and topologically related. Primitive information included in the tile directories, together with the tables, provides physical representations of geographical structures.

Types of information are used in VPF: entities (features), their locations and their properties (attributes), and their interactions (relationships). Each entry assesses a location and related properties, and is subject to various types of relationships with other entities. Self-describing metadata is also used to provide data and information.

Types of primitives: entity nodes (point features), edges (line and area features), connected nodes (line and area features), faces (area features), and text (text features). Primitive information is contained essentially in four types of tables: edge tables, ring tables, and face tables. Information for the single primitive, text, is stored in text tables. Each VPF table consists of a table containing metadata concerning table and column definitions, a row identifier, and

VPF, contiguity, direction and area definition are maintained through VPF's topology in the following manner:

stores references to its start node, end node and neighboring edges and faces stores the ID of its defining ring

- Each ring stores an associated face ID and a starting edge
- Each node stores its containing face

Figure 4 illustrates these topological concepts.

3.1.2. Production process. The VPF specification is used in the production of various database products such as Digital Nautical Chart (DNC) [4], World Vector Shoreline Plus (WVS+) [7], Vector Smart Map (Vmap) [6] and Urban Vector Smart Map (UVMap) [5]. These products are primarily used for GIS analysis, navigation and tactical operations. Intended uses of the products drive design decisions regarding accuracy requirements, and feature class and attribute inclusion, among others.

Many of the VPF products are derived from hardcopy sources such as maps and charts. For example, DNC is based directly on hardcopy Harbor, Approach, Coastal and General NIMA charts, each of which becomes a separate library in the digital product. Additionally, WVS+ includes information from Digital Landmass Blanking data, Operational Navigation Charts, Tactical Pilotage Charts, Joint Operations Graphics and others. Information from image sources is also used for some VPF products. NIMA provides an extraction guide for each VPF product that gives guidance on feature extraction issues such as minimum portrayal criteria, default attribute values, collection of text, and representation (point, line or area) of collected features, as well as numerous miscellaneous details.

Use of the extraction guidelines minimizes inconsistencies within a product by limiting the need for application of judgment calls during the extraction process. However, within the production of a single product, and certainly among the production of multiple products, random variabilities and perhaps even inaccuracies in the resulting product are inevitable due to the inherent inexact nature of the process. These present serious considerations for our work and represent much of the uncertainty for which some accounting must be made.

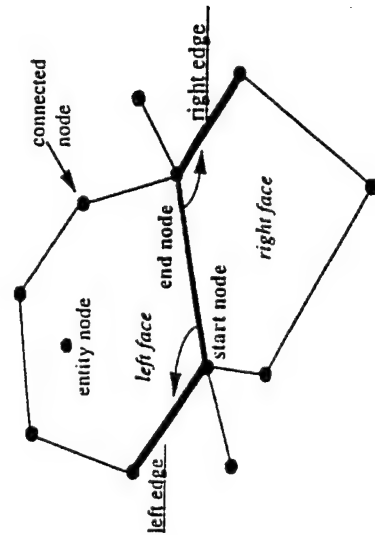


Figure 4. VPF's winged-edge topology.

ative regarding conflation for this project comes from the fact that many VPFs already exist and are in field use, while more are in the prototype stage and production. As noted previously, all VPF products are designed to assist performance of specific tasks such as navigation or mission planning. It is, however, and perhaps impossible, to produce a new digital product for every user. Furthermore, because the existing products are designed for specific goals, not necessarily with respect to properties such as scale and feature class attribute sets of VPF data concerned with goals other than those that specific products are designed to meet can benefit from having the "best" information for a given area as well as all possible VPF sources. This is a much more desirable approach than trying to produce a new VPF product for each newly discovered vector data need. For example, if analysis of a coastal region may be performed through combining, say, bathymetry information from DNC with vegetation information from VMap.

Feature matching

Feature matching is a process in which evidence must be evaluated and a conclusion drawn—not one in which equivalence can be determined. For example, fuzzy concepts such as "closeness" of two features, "similarity" of attributes and feature groupings are essential for determining whether all, if all feature pairs matched exactly, or deviated uniformly according to some measure. If, however, there would be no need to conflate the maps!

Feature matching can be considered as a type of classification problem. That is, to determine whether one feature belongs to the same "class" as another; in this case, the class is very restrictive—component members must be believed to be of the same type of problem can be handled through theories of evidential reasoning such as fuzzy logic [22] or Dempster-Shafer theory [21]. These theories provide likelihood measures for questions based on available, though not exclusive, evidence. For example, in feature matching, the question we must ask is: "Based on the available evidence, what is the likelihood (or probability) that a feature from map coverage 1 represents the same entity as feature B from map coverage 2?"

Fuzzy set theory is a theory of reasoning under uncertainty that is based on *fuzzy sets* [22]. It differs from traditional crisp sets in that elements are assigned a value by a membership function that represents the *degree* to which the element is believed to belong to a fuzzy set, rather than simply belonging or not belonging to a set. Elements may belong to multiple fuzzy sets with varying degrees of membership.

One of the problems with fuzzy sets is that they are not applicable to natural language concepts such as *large*, *small*, etc., where rigid set boundaries are difficult to establish. For example, a person would agree that a 6'6" male would belong to the *tall* set. Many, though fewer, would agree that a 6'2" male would also belong to the set. However, when the values fall to the 6'0" range, there would probably be much more disagreement about inclusion.

in the set. For traditional sets there would have to be a hard threshold, for example, 6'0", below which no elements would be included. However, it seems slightly illogical that a 6'0" male would be considered tall, while a 5'11.5" male would not. Fuzzy sets allow one to intuitively model situations such as this where some overlap between sets is a natural occurrence.

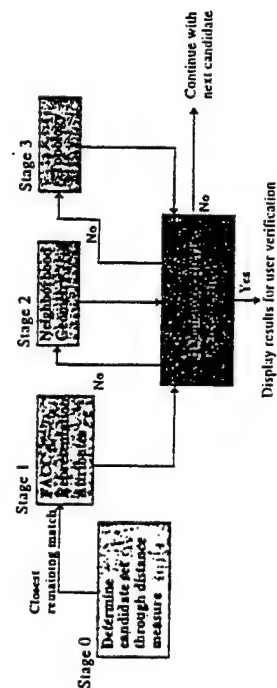
Dempster-Shafer theory is a method of inexact reasoning based on the modeling of uncertainty as a range of probabilities. It provides representations for the degree of belief in evidence (belief that supports a hypothesis), as well as the degree of disbelief in evidence (belief that refutes a hypothesis) and the nonbelief (neither belief nor disbelief). That is, it not only supplies results regarding the belief that two features are the same, but it also provides a representation of how much it is disbelieved that they are the same, along with the degree to which there is no evidence either way.

Our approach to feature matching draws from aspects of both fuzzy set logic and evidential reasoning. In particular, the assignment of matching scores for linguistic attributes is directly motivated by work in modeling linguistic variables by the use of fuzzy sets. For example, we need to be able to determine the semantic similarity of an attribute such as road surface type which may have a value of "hard" for one feature and "asphalt" for the potential matching feature. Obviously, the semantics of the two are not strictly equivalent, but neither are they contradictory. Likewise, the idea of combining scores from the different components of feature matching to arrive at a single matching score is very similar to techniques for the combination of evidence used in evidential reasoning. Details regarding our particular approach are provided in Section 4.

Our approach to feature matching employs a hierarchical rule base. For now, we will limit the discussion to feature matching between databases of the same or very similar scales. Figure 5 shows graphically the steps involved.

Each stage represented in Figure 5 contains a database of rules based on the set of criteria for that stage. These contain information from many sources, including the product extraction guidelines and data quality information, to name a few.

Before the feature matching process begins, a pre-processing step is performed to unify features which logically are parts of a single feature but have been segmented as part of the production process. For example, a single road in a VPF product may actually be represented by multiple line features, each comprising a segment of the road. Because



component features defined by a product for a geographic entity may vary from that given by a different product, we believe it is essential to our matching capabilities to identify cases such as this and unite the various feature sets into their logical composite before attempting to discern possible feature matches.

From the figure, the process begins at stage 0 by finding a candidate set of features from those in the complementary database that are geographically closest to the feature and the one being matched are then compared according to the Attribute Coding Catalogue (FACC) feature code (a five-character code identifies feature types), representation type (point, line or area) and value sets. At each stage in the matching process, a check is made to assess whether the match has been determined to a specified probability, e.g., 90%, based on the matching system. If so, the results are displayed and the user is allowed to accept, reject, or override them.

Part of the Digital Geographic Information Exchange Standard (DIGEST) [9] is a listing of common sets of features, their attributes and standardized product specifications use FACC for coding of product-specific features and within FACC, unique five-character alphanumeric codes are used to identify feature classes. The code is hierarchical in that the first character identifies the feature type, the second character identifies a subcategory, while the third through fifth characters identify features within the subcategory. For example, the feature code 'A111' identifies the specific feature type "timber yard" with the category A of culture category M of storage.

Features are uniquely identified by three-character alphanumeric codes, and codes ranging from 0 to 999 are used to represent attribute values. For example, the accuracy is represented by the code ACC and an accuracy value of *approximate* by the integer 2. In addition to encoded values, some attributes are allowed to be real, integer or text string values.

Products' use of the FACC feature codes is exploited in stage 1 of the feature matching process. Ideally, the first stage would be to find features with identical feature codes. If this is not possible, then we consider relaxing the matching constraint to include only the category and subcategory portions.

Category feature codes are determined to be the same or similar, then the set of their values is examined. The set of attributes can be different, and for the only category and subcategory matches are made, are almost certain to be different, often different features within a subcategory have overlapping sets of values. For example, within the *culture*, *storage* subcategory, many different feature types are identified: the attributes *Angle of Orientation*, *Aids to Navigation*, *Height Above Ground*, *Width* and more. For this process, we believe that strict equality of the feature codes is not used as matching criteria since the assignment during the production of the feature codes even the collection of such values beforehand is somewhat subjective. Use of fuzzy measures of equality are used.

The process continues with a more detailed analysis of the feature pair. Selected neighbors and their positions are investigated. Similarities or dissimilarities are considered as part of the matching process.

of the evidence for or against the hypothesis of equivalent features. Considering this first for entity nodes (points which do not represent the intersection of edges), we must investigate properties such as absolute position, positions (distance and direction) relative to nearby features that have already been determined to be matching features, as well as similarities in general neighborhood patterns. For connected nodes (nodes that represent the intersections of edges), geometric considerations include the directions and lengths of intersecting edges (including the spider function). Orientation and length of line features, as well as similarity in size and position of bounding boxes are used. Similar criteria are used for area features, in addition to the equivalence of values representing the contained area of each.

The final stage analyzes the topology of the two features at the lowest level of topology supported by the component databases. For products utilizing winged-edge topology, topological connectives of matching feature candidates can be checked for similarity. This can include, for example, left and right edges and left and right (adjacent) faces of edges for line and area features, or the containing face of point features.

Somewhat related to this subject is the work by Egenhofer, et al. [10] on the assessment of topological inconsistencies among multiple feature representations, i.e., features represented at various scales. The central premise of the framework presented in the paper is that topology, as first-class geographic information, must be preserved for any representation of the data, and that human identification of matching features relies more on the topological structure of the data than on its representation.

The framework is structured on the assumption of *monotonicity* of topological similarity under a generalization operation. That is, both the individual topology of each object and the topological relationships between objects must either remain unchanged or decrease in complexity when generalization is performed. The concepts of *object similarity* and *relation similarity* are used to determine topological consistency between levels of feature representations.

We believe that topological information, and in particular the idea of topological consistency, is a significant component in the determination of a solution to the feature matching problem.

Discussion so far has been based on a hierarchical approach of finding a candidate set and working against the candidate set under different constraints, such as attribute equivalence or similarity and neighborhood geometry. However, to insure the integrity of the feature matching process, an iterative procedure over the candidate set must be considered. The hierarchical rule-based approach considers the strongest criteria first and progresses to the weakest criteria to improve the likelihood of a match. Similarly, members of a candidate set must be considered iteratively to increase the likelihood of finding correctly matched features. This can be considered to be a part of the hierarchical process described above.

3.3. Feature deconfliction

While rubber-sheeting and feature matching processes deal with the difficult problems of map registration and identification of equivalent feature pairs, the next step is just as

it more—the resolution of the matched feature pair representations and one representation and one consistent set of attributes. *Feature deconfliction* is the process of unifying all parts of a matched feature pair into a single feature. Just as for the matching process, there is no simple solution to this problem. One of the considerations that must be taken into account include the respective source data at all levels (e.g., feature, coverage, library, product), cartographic editing during source production, and intended uses of both source products and the final product.

Table 1 discusses the three types of error in matching theory, as well as the methods for detection of each. These errors are:

False positive—a map feature is identified as having a match when in fact it does not.

False negative—a feature is correctly identified as having a match, but its matching feature was not selected; and

False match—a feature is identified as not having a match when in fact it does.

Table 2, the occurrence of a false negative is the most benign, and, as Saalfeld (1992) notes, it usually occurs in the early to intermediate stages before all matching criteria are checked. Occurrences of false positives and mismatches in the early stages are more serious because they can precipitate more errors and usually can not be corrected.

Table 3, to the three erroneous cases listed above, two possibilities for matching exist: (1) a feature's correct match is identified, in which case we call the match *correct*, or (2) a feature is correctly identified as having no match. Special consideration must be taken in determining the most appropriate way of dealing with this situation, which we call *one-feature*. The obvious options, of course, are to either include the feature from the final map or to include it as is.

Table 4, whether to include the feature or not, the factors of scale and intended use of the product must be evaluated. For example, feature collection criteria for VPF products include specifications for a minimum portrayal size below which no features are included, except under certain circumstances, such as extreme sparsity of features in the product, because each VPF product, as well as the conflated product, has an intended use and must consistently screen for features which would detract from, rather than enhance, the information content of the map based on its purpose.

Table 5, if a feature is included as a part of the resulting map, then the one-feature is added to the available information from the original map. However, there may be several issues involved. For example, consider the case of having neighboring features which are repeated features such that the information of the repeated features is a conflict. There is a possibility that the repeated-feature and the one-feature may coincide, intersect or overlap. Since the VPF standard [3] does not allow any features to coincide, intersecting edges without connected nodes, or overlapping faces, it is envisioned a case where new primitives must be created to support the winged-edge representation (Figure 11).

Our approach to resolving attribute and location conflicts for repeated features utilizes all available information regarding data quality for each product. The VPF standard [3] has provisions for data quality information at various levels of coverages and detail. The basic repository of such information is the *data quality table*. This table is used for storing both standard (within the table) and nonstandard (external) information. Data quality tables typically include information regarding source data, positional and attribute accuracy, including absolute horizontal and vertical accuracies, as well as information on the logical consistency, completeness and resolution of the data. Data quality tables are allowed at the database, library and coverage levels, depending upon the applicability of data quality characteristics to ranges of data. When such tables are defined at multiple levels, data residing at the lower level has precedence. Additional producer-defined files are also allowed as needed to document data quality information.

Lineage information is considered nonstandard, and as such is stored in an associated narrative file. This information is provided to document decisions made by the data producer that affect the fitness for use of the data. Examples include processing tolerances and rules concerning the interpretation of source materials. All lineage information available for the source is also included.

Data quality coverages are another way this type of information is provided. These coverages are composed of areas, each of which is assigned a specific set of quality characteristics through the use of attribute tables, standard data quality tables, and optional user-defined relational tables. Finally, data quality information can be provided at the feature level as attribute data.

Obviously, all of the provided data quality information is valuable in determining both appropriate attribute values and feature positions for the conflated product. These facilities can also be used to document the effects of the conflation process on the quality of the resultant data. However, in spite of the adequate provisions of VPF for documentation of such information, it is possible that individual products may not make extensive use of such facilities due to the fact that such information is simply not available. For example, one of the DNC product prototypes uses charts from the early 1900s and late 1800s as source materials. Quality of such antiquated data is difficult to analyze, to say the least.

Other problems with data quality information arise from the assignment process. For example, the VMap extraction guideline states that unless otherwise known, a default value of "accurate" should be assigned for data accuracy attributes. This is not an unreasonable solution to the issue of dealing with missing data; however, it does raise the question of how much faith can actually be placed in data quality information.

These two issues—how to deal with missing data quality information, and how much to trust the information that is present—are extremely difficult to resolve. In cases where data quality information is adequate, the process of feature deconfliction is an excellent prospect for the use of fuzzy or expert system techniques, similar to those employed for feature matching.

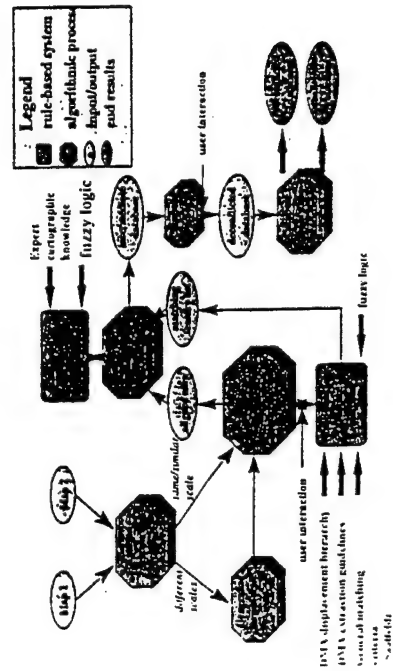
Deconfliction of attribute information for matched features is no trivial exercise. Many different kinds of discrepancies can occur, and often there is no obvious solution to the problem. Consideration must be made both for determining the set of attributes to support and for determining the values of the selected attributes.

dering simply the set of attributes related to each of the features, three cases
 the sets of attributes are equivalent, (2) one set is a superset of the other, or
 e disjunctive. Case 1 is trivial; for case 2, one might choose to support only
 es that both features have in common, or choose the superset of attributes.
 goal is to provide as much information as possible to the user, we choose to
 second option. Similarly, for case 3, we take the union of the sets.

al 2 raise another concern—that of determining the value to use for an
 ported by both features, but containing conflicting information. In order to
 e “best” value to use, attention must be made to items such as recency of
 accuracies and scale. For cases in which one feature’s information is clearly
 more reliable, accurate, etc.), that feature’s information is used for the
 ult. For the deconfliction of two features whose quality of information is not
 ined, a weighted algorithm is used which combines the disparate information
 eature “believability” factors. The method of combination varies according
 pe and meaning. For example, integers representing the lengths of a linear
 ply averaged, while integers representing discrete semantic values such as
 : type must be resolved differently.

of scale as it applies to attribute deconfliction is not at first obvious. The
 nes more apparent, though, in considering a feature such as a coastline
 all scale may have a soil type of unknown, but at a large scale may be broken
 atures, each with a specific soil type. The difficulty lies in determining where
 atent attribute values apply at varying scales.

ows the complete conflation system utilizing the approaches discussed in



review of conflation system.

3.4. VPF accuracy: theoretical principles

VPF products produced by NIMA go through various phases from initial survey to release. As expected, each such phase allows the opportunity for inaccuracy to be introduced, whether by instrumentation or human factors. *Digitizing the Future* [2] states that, to make these errors known to the customer, statistical error probability values based on error distributions are supplied, thereby providing a probable maximum displacement of NIMA's estimated position from true position and a level of confidence for the product's accuracy.

NIMA defines absolute horizontal accuracy in what Digitizing the Future [2] refers to as Circular Error Probability (CEP). The classical definition of CEP can be thought of in terms of a missile being fired at a target (see, for example, [15]). Naturally, variations in two orthogonal directions can occur, and these deviations manifest themselves in two random variables X and Y , often considered to follow a bivariate normal distribution. Forming $R = (X^2 + Y^2)^{1/2}$ yields a random variable commonly known as *radial error*. Its distribution function $F_R(r)$ yields the probability of a missile falling within a circle having the target as center and radius r . One traditionally defines CEP as the median of this distribution, namely, that value of r for which $F_R(r) = 0.5$ (i.e., "the probability of the missile falling within a distance of r from the target is 50%"). NIMA, however, does not restrict the definition to 0.5; this is explained fully in a subsequent section on precision indexes.

To interpret VPF CEP in these terms, consider the true position of a point (an unknown value) as the missile impact point and NIMA's estimate of this true position as the missile target. A horizontal accuracy of 25 m at 95% CEP therefore translates to the following: Given 100 points of NIMA-specified geographic coordinates, at least 95 are within 25 m of their true position.

As an aside note, a one-dimensional $100(1 - \alpha)\%$ confidence interval is an interval estimate of a scalar, and this estimate has a similar definition as the CEP. In fact, CEP appears to be the two-dimensional analog. However, a common misconception of an interval estimate, say $[a, b]$, of a scalar, say, μ , is "the probability that $\mu \in [a, b]$ is $1 - \alpha$." In actuality, the probability that, $\mu \in [a, b]$ is 1 or 0, i.e., inside or not. The correct interpretation is "if one properly samples and constructs other interval estimates in the same manner, approximately $100(1 - \alpha)\%$ will contain the value μ ." Herein lies the difference: With CEP, the circular error yields more than an interval estimate of NIMA's estimated positions.

3.4.1. Types of measurement errors. To understand the interpretation of NIMA's horizontal accuracy in the context of conflation, one must first understand the concept of error. Measurement errors may be classified generally as (a) blunders, (b) systematic, or (c) random. *Blunders* are the easiest to comprehend, since they are aptly named. These mistakes are usually "large" and attributed to human error such as transposing digits, careless observations, and miscalculations. *Systematic* errors are those errors associated with measuring devices and almost always follow fixed patterns. What remains after blunders and systematic errors are the *random* errors. These are characterized in terms of

If a sequence of measurements of a given quantity are made, and the separate measurements are nonpredictable, then the error is termed a specialized error as discussed by Rabinovich [18].

Allocation of measurement errors is important, since each is handled differently. As NIMA provides no information with regard to systematic errors, blunders, we consider only the random component in the following

ular error. The random variable $R = (X^2 + Y^2)^{1/2}$ defined previously will be shown (see [14]) that the probability distribution function $F_R(r)$ is in the following form.

$$= (\sigma_x \sigma_y)^{-1} \int_0^r \rho e^{-(\rho^2/4\sigma_x^2)(1+\sigma_y^2/\sigma_x^2)} I_0 \left[(\rho^2/4\sigma_x^2) \left((\sigma_y^2/\sigma_x^2) - 1 \right) \right] d\rho \quad (1)$$

where the integration is over $[0, r]$ and I_0 is the modified Bessel function of the first kind, zero order, since NIMA gives no indication that the two directions have inherently different accuracies, we may consider $\sigma_x = \sigma_y = \sigma$ and use

$$= 1 - e^{-r^2/2\sigma^2} \quad (2)$$

where the deviation value σ is termed *circular standard error*, a precision index and r measures on error dispersion.

Position indexes. Greenwalt and Schultz [14] define four precision indexes for positions associated with circular distributions. These values provide an error index which will not be exceeded with a certain probability. They are the *circular standard error* σ , the *circular probable error* (CPE) or *circular map accuracy standard* (CMAS), and the *circular near-certainty error*, called *three-five sigma*.

The standard error can be derived from equation (2) simply by substituting σ for r , which results in a value of approximately 39.35%. Three-five sigma is found computationally by using 3.5σ for r , which results in a circular probability of 99.78%; hence the alternate name *circular near-certainty error*. CEP, as mentioned previously, represents that value for which half of all circular distribution will not exceed. CMAS, based on the percentage level used in National Map Accuracy Standards [14] represents an error beyond which 90% of defined points will not exceed.

These definitions and *Digitizing the Future* [2], VPF is clearly using CMAS as a precision index. However, to be consistent with NIMA's definition, we will not reserve a term for the term CEP.

Combinations of measurements. With the knowledge of the circular distribution of error, we now turn our attention to interpreting the accuracy of measurements.

Data conflated from various VPF products can theoretically be viewed as that part of metrology dealing with combining the results of measurements. In *Measurement Errors: Theory and Practice* [18], Rabinovich considers the situation involving a quantity A ; L groups of measurements of A ; unbiased estimates of A , p_1, p_2, \dots, p_L , from each group; where variances of the measurement in each group as well as the number of measurements in each group are known. The problem of finding an estimate of A using the estimates from each group has a mathematically rigorous solution. The solution involves a linearly weighted sum of the estimates (a *combined average*) $\bar{x} = \sum^L w_i p_i$ where the weights may be based on the number of measurements or the variances within the groups. The variances of the groups are also used in defining the variance of the combined average. Certain exceptions (which will not be considered here) are made when the group variances are unknown.

In the VPF context, the NIMA-specified measured positions originate from the radial error distribution and are considered as estimators of true locations. Although we do not know specifically how NIMA arrived at the estimate, whether by taking several measurements over time and computing a centroid or using estimates from a variety of sources, by its own definition NIMA guarantees the estimators will follow a circular distribution.

Consider the example of conflating points from two products, one with horizontal accuracy of 75 m at 95% CEP, the other, 25 m at 90% CEP. Figure 7 provides a representation of two estimates p_1 and p_2 of the same feature A , whose true location is, of course, unknown. The corresponding estimators are denoted P_1 and P_2 , and the conflated estimator is P , where P is a function of P_1 and P_2 . (We are using *estimator* in this context as a synonym for random variable.)

Several questions become immediately apparent. Based only on absolute horizontal accuracy measurements and the estimates p_1 and p_2 , how does one choose the conflated estimate p ? More importantly, what accuracy is associated with p ? Does one weight more

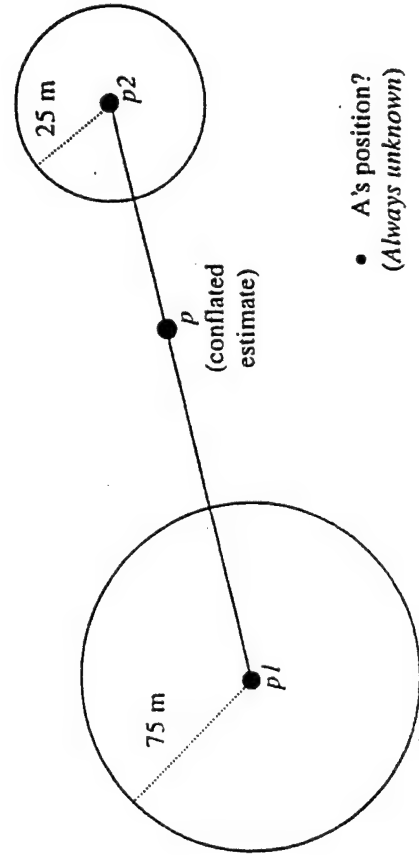


Figure 7. Two point representations of the same feature A .

stan

estimate representing the "smaller circle/variance" (25 m) or the estimate of a "circular confidence" (95%)? Based on its status as a function of P_1 and P_2 , does the error distribution and thus have an associated CEP?

It's "proof" of how to choose a combined estimate indicates that, if $\text{Var}(P_1) = \sigma_1^2$ and $\text{Var}(P_2) = \sigma_2^2$ are known, the weights can be selected as $w_1 = (1/\sigma_1^2)/s$ and $w_2 = (1/\sigma_2^2)/s$, where $s = (1/\sigma_1^2) + (1/\sigma_2^2)$. Due to this inversion we have that the estimate of greater variance is given the lesser weight in the estimate. Moreover, since the weights are nonrandom quantities, one may consider properties of variance, that the variance of the conflated estimator is $1/s$. If only estimates of group variances are known, then the weights are based on measurements taken within each group, $w_i = n_i/N$, where $N = \sum n_i$. Finally, of "smaller variance" versus "greater confidence" is in a sense misleading, when the case with VPF products, the same confidence level is specified for

of circular distribution, we may determine from equation (2) the values of σ_1 and P_2 , under the assumption that the estimators P_1 and P_2 are independent. As in *Digitizing the Future* [2] that "Because of the way DMA processes location, the location and elevation of each data point can be assumed to be equally measured." With respect to the combined estimator P , Greenwalt and allude to the fact that combining independent variables of radial error will yield circular error resulting from errors associated with each variable. Consider the previously posed problem of estimated locations $p_1 = (x_1, y_1)$ at P_1 and $p_2 = (x_2, y_2)$ at 25 m/90% CEP. Using equation (2) we have that

$$-r^2/2 \ln(1 - F_R(r)) \quad (3)$$

is the natural logarithm function) from which it can be determined that $\sigma_1^2 \approx 135.72$. Using these values to form the weights yields $w_1 \approx 0.13$ and the conflated estimate would be $p \approx (0.13x_1 + 0.87x_2, 0.13y_1 + 0.87y_2)$. The error of the conflated estimate p at 90% CEP, since $\sigma^2 \approx (135.72)^{-1} - 1$, we may use equation (3) and solve for $r \approx 23.37$. As a pictorial representation of this example.

Figure 9 illustrates one way in which the combination of information from two sources can lead to "better" information, at least in terms of positional accuracy. The knowledge of absolute horizontal accuracy requirements for both products allows us in some cases to refine the circular error for the conflated result to a lesser than that of the lesser circular error of the components—a counterintuitive result. Of course, accuracy cannot be changed without changing the instrument.

When all VPF horizontal accuracy values are based on 90% CEP, one may see the case of figure 9 without regard to the 0.9 probability to make some observations. In any of these cases, one would feel justified in choosing the point to lie on the line segment connecting the two estimates. Although part of the problem with one or both sources since the error limits do not

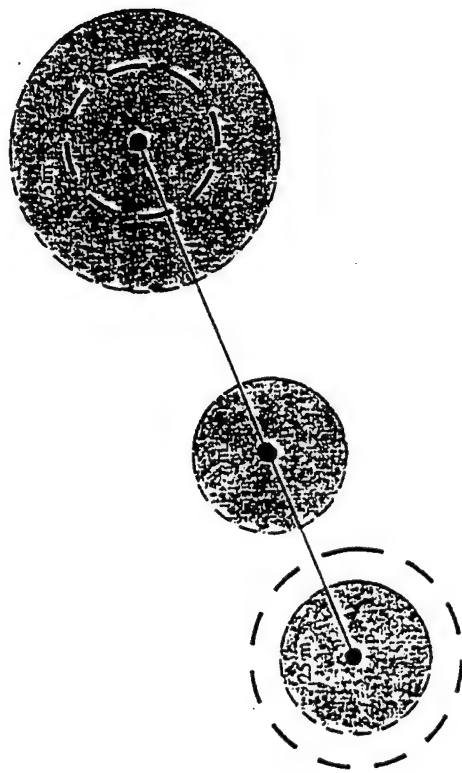


Figure 8. Determining r such that $F_p(r) = 0.9$.

overlap, this case is legitimate due to the possibility of the true location lying outside both circular buffers. Of special note, part *b*, in which the circles are tangent, tends to indicate that the point of tangency should be the conflated point. Indeed, if the circular limits were 100% CEP (an unrealistic expectation), then the true location would be the point of tangency. The remaining cases have interpretations which yield information on selection

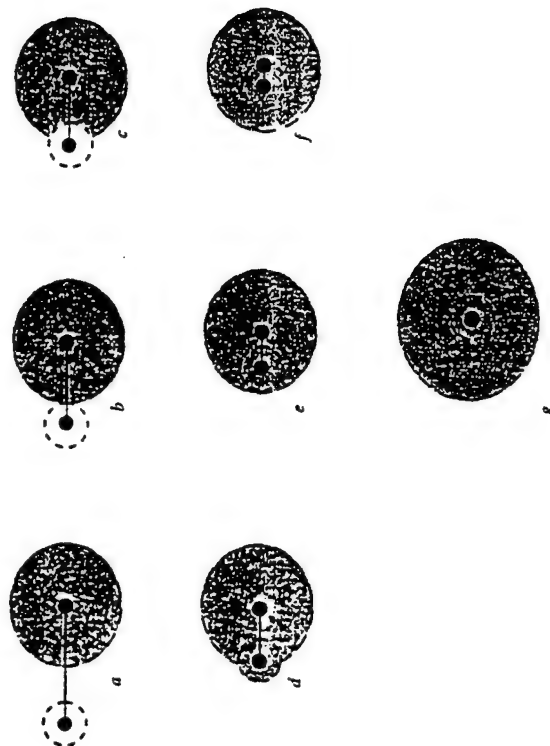


Figure 9. Relationships between two estimates of a point feature and their corresponding circular error.

ics of the domain and the linguistic terms. The characteristics of the similarity are:

$$s_i(y, x) = s_i(x, y) \quad \text{symmetric}$$

ies $x, y \in \text{domain of attribute } A$. For well-defined values of the domain (e.g., own" or "other")

$$s_i(y, y) = 1 \quad \text{reflexive}$$

a well-defined value.

ample, we consider the Railroad attribute *RRA* (Railroad Power Source), restricted to the values (0—Unknown, 1—Electrified track, 3—Overhead 4—Non-electrified, 999—Other). The similarity table for *RRA* is shown in e linguistic similarity is symmetric we need only show the lower triangular e table. Note that since 1, 3, and 4 are well-defined linguistic terms they are he reflexive value of 1 on the diagonal, e.g., $s_{RRA}(3, 3) = 1$. However, since 0 : non-specific categorical values for this particular domain, their diagonal determined to be less than 1, reflecting the potential lack of exact matching for ic terms.

ost features have more than one attribute, we must also consider semantic hips among the attributes in determining matching features. These are s rules in the expert system which return associated weights. These weights ither add credence to the hypothesis of matching features, or to weaken it. As consider the Railroad attributes *LTN* (Number of tracks) and *RRC* (Railroad ie rule for the relationship between the two attributes is expressed as:

$$R1.ltn = 3 \text{ and } R2.ltn = 2) \text{ and}$$

$$R1.rtc = 16 \text{ and } R2.rtc = 16))$$

$$W_{ra} \leftarrow 1.0$$

$$W_{ltn} \leftarrow 0.5,$$

represents the first Railroad object and *RR2* represents the second. This rule conflict in the values of the length attribute of the two features. We see from e that the resulting weight for *LTN* is weakened, giving it less influence than in the combined matching score.

Similarity table for attribute *RRA*.

	0	1	3	4	999
0.2					
0.2		1			
0.2		0.6	1		
0.2		0.1	0.1	1	
0.2		0.2	0.2	0.2	0.2

A composite matching score is then computed from the combination of the expert system weights and the similarity table values. This score is given as:

$$MS_{i,j} = \left(\sum_{k=1}^N [\text{sim}A_k(F_i, F_j) \times ESW_{Ak}] \right) / N$$

where

A_k = k th attribute in both F_i and F_j , where $0 \leq k \leq N$.

N = number of attributes that describe both F_i and F_j .

ESW_{Ak} = weight associated with A_k computed by the expert system.

4.3. Geometric analysis

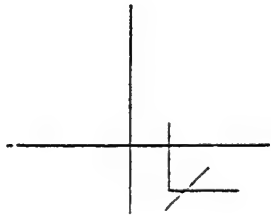
The second part of our implementation work consists of a technique for qualitatively matching linear features on the basis of shape similarity. This broadens our basis for feature matching by adding a spatial component to the non-spatial attribute matching algorithm presented above.

The first step involves a pre-processing of the features to bring them into a standard position for the shape similarity analysis. Translation, rotation, and scaling transformations are applied in sequence to accomplish this positioning. Translation is performed by moving the features' starting nodes to the origin of planar axes. Then, the end nodes are rotated to the x-axis. Finally, the features are scaled so that the starting and ending nodes are equal to [0, 0] and [1, 0] respectively. These transformations allow us to more reliably compare linear features of different sizes. Figure 10 shows the results of standardizing two linear features through this process.

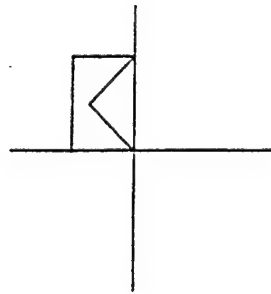
After the features have been standardized, a process called *node merging* takes place [20]. This requires all nodes from one feature to be mapped onto the other feature, and vice versa. This is performed on the basis of a node's distance from the starting node of the feature. Consider two features, F_1 and F_2 , comprised of nodes $(n_{11}, n_{12}, \dots, n_{1j})$ and $(n_{21}, n_{22}, \dots, n_{2k})$, respectively, and where the associated ratios are given as $(r_{11}, r_{12}, \dots, r_{1j})$ and $(r_{21}, r_{22}, \dots, r_{2k})$ such that $r_{11} = r_{21} = 0.0$ and $r_{1j} = r_{2k} = 1.0$, where r_i is the ratio of the distance from start to node n_i to the overall length of the linear feature. Then, new features F_1 and F_2 are created that each have the same number of nodes such that each node is the result of merging the features' nodes based upon their ratios.

Given that the features are in standard position and the nodes have been merged, we begin the last phase in assessing shape similarity. This involves determining a normalized distance measure between corresponding nodes of the two features, given as

$$SHsim = \left(\sum_{i=1}^{j+k-2} \text{Dist}(n_{1i}, n_{2i})(D_1 + D_2) \right)$$



(a)



(b)

near features (a) before and (b) after standardization process.

$n = 2$ is the number of distances to be computed, and D_i is the total length of lines that features with a higher shape similarity measure are more likely to matching features. It is based on Frechet's measure of distance, L_2 - between polygonal arcs. Suppose that we have two functions

$$1] \rightarrow R^2$$

$$1] \rightarrow R^2$$

p_1, \dots, p_n and q_0, q_1, \dots, q_m are ordered point sets for P and Q . Then the set of ratios used for computing the distance between P and Q at all computed as described for node merging above. Thus, for Frechet's measure, $\frac{1}{2}$ is used.

$$\left(\int_0^1 \|P(r) - Q(r)\|^2 dr \right)^{1/2} \left(\sum_{i=1}^{n+m} \int_{r_{i-1}}^{r_i} \|P(r) - Q(r)\|^2 dr \right)^{1/2}$$

then that (x_{Pi}, y_{Pi}) and (x_{Qi}, y_{Qi}) are the coordinates of $P(r_i)$ and $Q(r_i)$, and $\Delta_{x,i} = x_{Pi} - x_{Qi}$ and $\Delta_{y,i} = y_{Pi} - y_{Qi}$ are the measured separations of the lines at the bending points of either of the lines, then the closed form of the L_2 - Distance is represented by

$$\sum_{i=1}^{n+m} \left((r_i - r_{i-1}) (\Delta_{x,i-1}^2 + \Delta_{x,i-1} \Delta_{x,i} + \Delta_{x,i}^2 + \Delta_{y,i-1}^2 + \Delta_{y,i-1} \Delta_{y,i} + \Delta_{y,i}^2) \right)^{1/2}$$

the mathematical exposition presented here presents the objective portion of

feature matching. The subjective portion represented by the expert system is much more difficult to capture. In response to this problem, Foley et al. [11], [12] present a web-based knowledge acquisition tool for capturing expert knowledge needed for conflation.

5. Summary and future work

In this paper we presented an approach for the conflation of attributed vector data that utilizes both spatial and nonspatial properties for feature matching and deconflation. A rule-based paradigm is used which incorporates properties of uncertainty inherent in the conflation problem. We have given a technique for feature matching based on semantic similarities of attribute values and shape similarity of linear features. An implementation based on an expert system and the OVPF prototype and utilizing NIMA's VPF data has been performed which demonstrates the effectiveness and practicality of the method.

Additionally, a detailed study of the effect of conflation of point features from VPF products showed a process for selecting the conflated point which had a smaller variance than either of the source points. Though specific to VPF, this study could be easily extended to other vector data, due to common practices concerning map accuracy standards.

Remaining work includes full implementation of the feature matching process shown in figure 5, including an expanded set of attribute similarity tables and functions, as well as neighborhood geometry and topology matching techniques.

Table 2 summarizes the identified issues for conflation of VPF products and the approaches presented in this paper.

Table 2. Conflation of VPF Products.

Issue	Approach
Feature Matching	Automated, iterative matching process based on distance, FACC, representation, attributes, neighborhood geometry and topology. Implemented as a rule-based system augmented with information from product extraction guidelines, specifications and displacement hierarchy as well as a logic for reasoning under uncertainty.
Alignment	Triangulation of participating coverages followed by iterative alignment based on edges and nodes.
Deconflation of attributes	Select "best" attribute set and values based on data quality. Implemented as a rule-based system containing expert cartographic knowledge and fuzzy logic capabilities.
Deconflation of representation	Weighted interpolation based on accuracies and scales of sources and desired output.
Analysis of results	Audit trail of all decisions maintained during feature matching, alignment and deconflation. User allowed to view history of decisions and explanations for all rule-based processing and view all source data in its original form. General guidelines for accuracy and fitness for use generated and exceptions noted.

Comments

I thank the Office of Naval Research, and specifically the Command and Control under the direction of Mr. Paul Quinn, for funding this work. We also wish to thank Breckenridge, Naval Research Laboratory, and Barbara Ray, Planning Inc., for their helpful comments in preparing this paper.

5

1. M. Cobb, D.K. Arctur, and K. Shaw. "An Object-Oriented Approach for Handling Topology in 'Shells.'" in *Proc. GIS/LIS 95*, Nashville, TN, 163-174, 1995.
2. Mapping Agency (DMA). Digitizing the Future. Fourth Edition. DMA Stock No. GITALPAC.
3. Mapping Agency (DMA). Military Standard: Vector Product Format, Draft Document No. MIL-7. Defense Mapping Agency, Fairfax, VA, 1993.
4. Mapping Agency (DMA). Draft Military Specification for Digital Nautical Chart, MIL-D-89023. Mapping Agency, Fairfax, VA, 1994.
5. Mapping Agency (DMA). Draft Military Specification for Urban Vector Smart Map (UVMMap). MIL-U-89035. Defense Mapping Agency, Fairfax, VA, 1994.
6. Defense Mapping Agency (DMA). Military Specification for Vector Smart Map (VMap) Level 0, MIL-V-7. Defense Mapping Agency, Fairfax, VA, 1994.
7. Mapping Agency (DMA). Draft Military Specification for World Vector Shoreline (WVSPPLUS). MIL-7A. Defense Mapping Agency, Fairfax, VA, 1995.
8. J. and U. Rodney. "Automatic Conflation of Digital Maps," in *Proc. IEEE-IEE Vehicle Navigation and Systems Conference*, A27-A29, 1993.
9. Geographic Information Working Group (GIWG). The Digital Geographic Information Standard (DIGEST), Directorate of Geographic Operations, Department of National Defense, 1994.
- 10.hofer, E., Clementini, and P.D. Felice. "Evaluating Inconsistencies Among Multiple Versions," in *Proc. 6th Intl. Symp. on Spatial Data Handling*, Taylor & Francis, 1994.
11. Petry, M. Cobb, and K. Shaw. "Utilization of an Expert System for the Analysis of Semantic Data for Improved Conflation in Geographic Information Systems," in *Proc. of the 10th Intl. Industrial and Engineering Applications of AI*, Atlanta, GA, 267-275, 1997.
12. Petry, M. Cobb, and K. Shaw. "Using Semantic Constraints for Improved Conflation in Spatial Data," in *Proc. 7th Intl. Fuzzy Systems Association World Congress*, Prague, 193-197, 1997.
13. Lian. "Triangulations for Rubber-Sheeting," in *Proc. Auto-Carto 7*, Falls Church, VA: ACSM/ASPRS, 1995.
14. J. and M.E. Shultz. "Principles of Error Theory and Cartographic Applications," ACIC Report No. 96, United States Air Force Aeronautical Chart and Information Center, St. Louis, MO, 1981.
15. J. and N.L. Johnson, (Eds.). Encyclopedia of Statistical Sciences. New York: Wiley, 1989.
16. B.J. Schachter. "Two Algorithms for Constructing a Delaunay Triangulation," *International Journal of Computer and Information Sciences*, 9(3):219-241, 1980.
17. J. and A.J. Saalfeld. "Conflation: Automated Map Compilation—A Video Game Approach," in *Auto-Carto 7*, Falls Church, VA: ACSM/ASP, 1985.
18. J. and A.J. Saalfeld. "Measurement Errors: Theory and Practice," Woodbury, NY: Wiley, 1995.
19. J. and A. Saalfeld. "Match Criteria for Automatic Alignment," in *Proc. Auto-Carto 7*, Falls Church, VA: ACSM/ASP, 1985.


- A. Saalfeld. "Collation: Automated map compilation," *Int. Journ. GIS*, 2(3):217-228, 1988.
- G. Shafer. *A Mathematical Theory of Evidence*, Princeton University Press, 1976.
- L.A Zadeh. "Fuzzy Sets," *Information and Control*, 338-353, 1965.





La Cobb is an assistant professor at the University of Southern Mississippi. Her primary research interests are distributed object-oriented geographical information systems, the development of object-oriented techniques to improve digital mapping databases, and the application of fuzzy logic and evidential reasoning techniques to spatial data modeling and spatial reasoning systems. She received a B.S. in computer science in 1987 from the University of Southern Mississippi, and M.S. and Ph.D. degrees in computer science from Tulane University in 1993 and 1995, respectively.



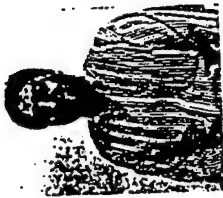
I Chung is an electronics engineer at the Naval Research Laboratory where she develops object-oriented techniques to improve digital mapping databases. She has experience in object-oriented programming and object-oriented database management systems. She is interested in the distributed communication of mapping data to base servers and tactical users in real time using Common Object Request Broker architecture. She received a B.S. in electrical engineering at University of Maryland at College Park and M.S. in electronics engineering from George Washington University.



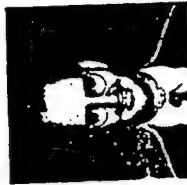
is a computer scientist at the Naval Research Laboratory. His primary research interests include fuzzy reasoning, and developing Internet-based applications. He received an M.S. in computer science from Tulane University in 1995 and a Ph.D. in computer science in 1997, also from Tulane.



received a Ph.D. in computer science from Ohio State University in 1975 and is a Full Professor of Electrical Engineering and Computer Science at Tulane University, New Orleans, LA. He is the Director of the Center for Intelligent and Knowledge Based Systems (CIAKS) at Tulane and has research interests in intelligent information systems. Dr. Perry has over 170 scientific publications written or edited. He is associate editor of IEEE Transactions on Fuzzy Systems. He was also a member of the 1996 IEEE International Conference on Fuzzy Systems and has received the K.S. Fu Fuzzy Sets Society and several IEEE awards. He was elected an IEEE Fellow in 1996 for his contributions to decision modeling in databases.



Kevin Shaw is an electrical engineer at the Naval Research Laboratory where he is the team leader for the Digital Mapping, Charting, and Geodesy Analysis Program (DMAP). He has over 13 years of experience in the digital mapping field. He received a B.S. in electrical engineering at Mississippi State University in 1984, an M.S. in Computer Science at the University of Southern Mississippi in 1987, and an M.E.E. in electrical engineering at Mississippi State University in 1988.



Vincent Miller is a Research Scientist with the Spatial Analysis Research Laboratory, located in the Tulane University Medical Center School of Public Health and Tropical Medicine. He has nine years of experience with the Naval Research Laboratory in the digital mapping field. Currently, he is interested in geographic information systems and their application to environmental health research. He received a B.S. and an M.S. in mathematics from the University of Southern Mississippi in 1987/89 and an M.S. in mathematics from Tulane University in 1992.

IEEE

May/June 1998

SOFTWARE

Building the Community of Leading Software Practitioners

Webward Ho!

Also

Selling OO:
Who's Buying, Who's Not

Y2K B42L8

How to Build
More Usable APIs



IEEE
COMPUTER
SOCIETY

<http://computer.org>

The authors faced the dual challenge of first converting a relational database to the OO paradigm, then migrating it to the Web. Here they describe the procedures and technologies they used.

An OO Database Migrates to the Web

Marie A. Cobb, University of Southern Mississippi
Harold Foley III, Ruth Wilson, Miyi Chung, and Kevin B. Shaw, Naval Research Laboratory

In the past several years, object-oriented methods have found increasing application in software engineering. OO's advocates claim that the paradigm's features—such as inheritance, encapsulation, and polymorphism—offer programmers a flexible and naturalistic means of tackling complex software design problems. More recently, traditional databases have been restructured to take advantage of OO methods. Java's advent enhanced OO's benefits by providing a vehicle for running such applications across several platforms without the need for extensive cross-platform coding.

Currently, many organizations face the challenge of migrating their legacy data to smaller, more modern applications that can be distributed throughout an organization via the Web or a company intranet. The National Imagery and Mapping Agency is one such organization. NIMA is the principal and often sole supplier of mapping data for the US Department of Defense.

Originally, mapping data took the form of paper charts, maps, and satellite photos. However, in the 1980s NIMA began the tedious process of transforming their paper products into a more usable and timely digital format. Vector Product Format (VPF),¹ a relational data model, became the digital standard for disseminating NIMA's charting and mapping data. The Digital Mapping, Charting and Geodesy Analysis Program (DMAP) at the Naval Research Laboratory, Stennis Space Center, Mississippi, was initiated to perform reviews and provide feedback on NIMA's new digital mapping prototypes.

In 1991, these reviews revealed that the relational format chosen for the data model was cumbersome and less than optimal in several ways. The problems stemmed from the difficulty in representing complex geographic data by decomposing it to fit the flat-file structure imposed by the data model. The DMAP team proposed an object-oriented data model as a possible alternative, and in 1994 secured funding to develop an OO prototype of the Digital Nautical Chart, one of the most complex VPF products. The OO prototype, Object Digital Nautical Chart, proved successful, and the team went on to develop object models and prototypes for three other VPF products, World Vector Shoreline Plus, Vector Smart Map, and Urban Vector Smart Map. The more general prototype, known as Object Vector Product Format, could import any of NIMA's VPF products distributed on CD-ROM, then convert the data into an object format for display, query, and updating purposes.

OVPF continued to evolve over the next two years through funding from the Office of Naval Research and NIMA. What had been a purely vector-based system expanded to include OO models for NIMA's two other families of digital products, Raster Product Format and Text Product Standard. RPF is the basic format for all raster products such as satellite imagery and scanned charts, while TPS provides an SGML-based standard for distribution of textual information such as sailing directions.

PLANNING FOR THE WEB

With the completion of the Object Digital Nautical Chart component in 1995, team members began contemplating future directions for the project. Given NIMA's role as data distributor, and considering the implications of their newly formed Global Geospatial

Information and Services modernization program, we knew that electronic dissemination and remote updating of mapping data should be part of future plans for the prototype. We considered several architectures, including the Common Object Request Broker Architecture and more static, Web-based methods. A demonstration in the summer of 1995 showed a simple remote updating capability based on a Web browser, Perl scripts, GIF images, and a remote human operator.² Although this was a start, we knew that we wanted a much more sophisticated level of distributed operation, and the team began searching for the right direction to take to provide network-based functionality.

At that time, the Object Management Group's Corba seemed the most promising architecture for object-oriented and standard systems interoperability. Java also made its debut in 1995, awing developers with powerful new Web-based capabilities. While the team watched these new developments closely, investments in these two areas were not viable at the time, given that Corba had not yet been accepted as a standard and it was not readily apparent that Java's ultimate popularity would match its promise. Therefore, we decided not to pursue the issue at that time, but to carefully monitor these evolving technologies while proceeding with application expansion in other areas.

OPPORTUNITY KNOCKS

During this period, the team became involved in another project funded by NIMA's Defense Modeling and Simulation Office and Terrain Modeling Program Office. The first phase of this project involved the development of a new VPF product standard and prototype, tailored for use by members of the modeling and simulation community. This effort resulted in the Extended Vector Product Format (EVPF),³ which, among other things, allowed the depiction of triangulated irregular networks to represent terrain data in three dimensions.⁴ Using OVPF, project staff developed an OO model for EVPF and incorporated a prototype data set, Modeling and Simulation Extended Vector Product, into the system.

Concurrently with this work, members of the DoD modeling and simulation community devel-

Our project's goal was to create a Java-based mapping client.

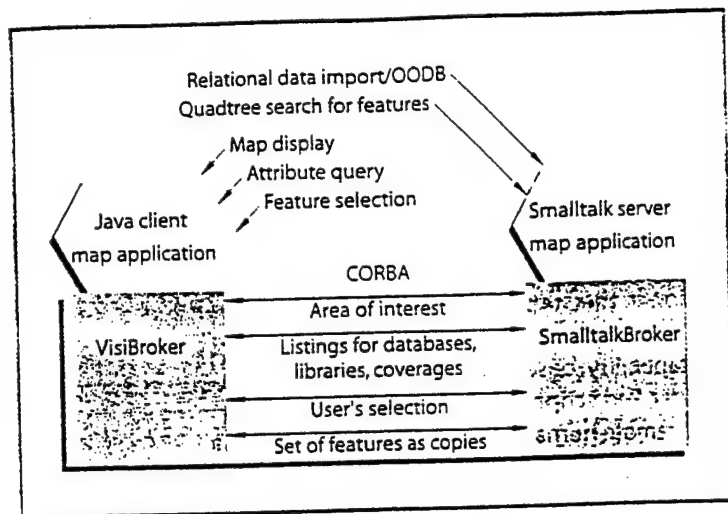


FIGURE 1. Basic system design for the Web-based OO mapping database.

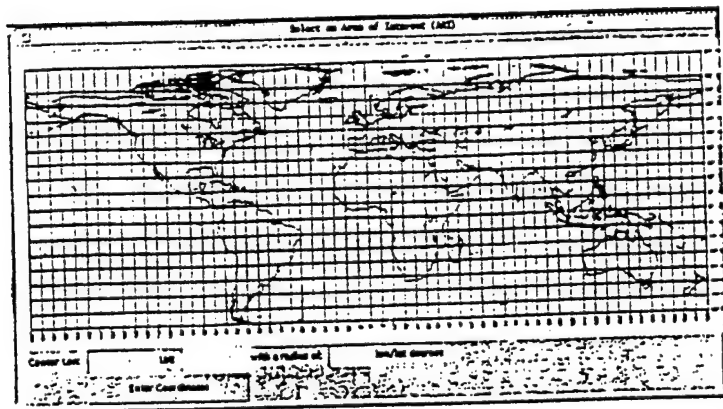


FIGURE 2. The mapping database's world map. Users can select an area of interest from this map by clicking on a location or typing coordinates that form a bounding rectangle.

oped a conceptual model for exchanging synthetic environment information. When completed, the Synthetic Environment Data Representation and Interchange Standard model (SEDRIS) will let heterogeneous modeling and simulation systems exchange information using APIs.

In early 1997, the NIMA EVPF program manager requested that the NRL team research available OO technology for the transfer of modeling and simulation data across a network as a SEDRIS support subsystem. At that time, Corba 2.0 emerged as a fairly stable standard with promised vendor support. Meanwhile, Java exhibited a meteoric rise as the OO programming language of choice. Given these trends, we felt secure in using these supporting technologies for a full-scale migration of our ap-

plication to the Web. The "Supporting Technologies" boxed text on pp. 28-29 provides more information on the Corba standard and the Smalltalk and Java programming languages.

SYSTEM DESIGN

The basic architecture of the system is shown in Figure 1. Our project's goal was to create a Java-based mapping client that would provide display and query capabilities for a set of geographic objects (features) that would be retrieved from the Smalltalk mapping prototype acting as a server. We planned to base communication on the Corba specification. We also planned to keep the remote fetching of objects completely transparent to the end user, who would be able to manipulate the features as if they were local.

The retrieval setup we decided upon provided a world map, shown in Figure 2, from which the user could select a particular area of interest. The user could choose an AOI by either selecting a location on a world map, or by manually entering coordinates to form a bounding rectangle.

From this input, the application selects and transmits a bounding box of the geographical region selected, and the Smalltalk server responds with a set of OVPF database names, as Corba string objects which contain data for that region. Once the user selects a particular database from the returned set, the system likewise returns the set of pertaining library names. Upon selecting a particular library, the system retrieves a list of coverage names. Finally, after the user chooses a coverage, the system returns the set of feature class listings. All this interaction invokes the appropriate server methods for determining geographically relevant responses. Finally, the client retrieves from the server all the features specified by the user. As opposed to previous communications between the client and the server, features returned at this stage are complex objects. These feature objects contain both geometric (x, y, and z coordinate) information and attribute information. Thus, the features can be displayed both graphically and textually and can be manipulated in ways similar to those of server-based operations.

Figure 3 shows the Java client map GUI, along with a set of hazard area features that were retrieved from the server application. Along the left side of the map is the set of list panes from which the user selects—and the server alternately supplies—listings for database, library, and coverage sets.

tions. As Figure 3 shows, users may display, zoom in, and zoom out the feature objects.

Additionally, users can request a query window, which appears as a pop-up window beside the map. The query window displays the list of feature IDs, from which the user may select for display associated attribute information. The Select button allows users to highlight the feature on the map that corresponds to the selected feature ID in the query window.

We loosely based the client map design on the map design used in the server application. The principle difference at this time is that no editing functions are available with the client map—it is strictly for visualization and feature querying.

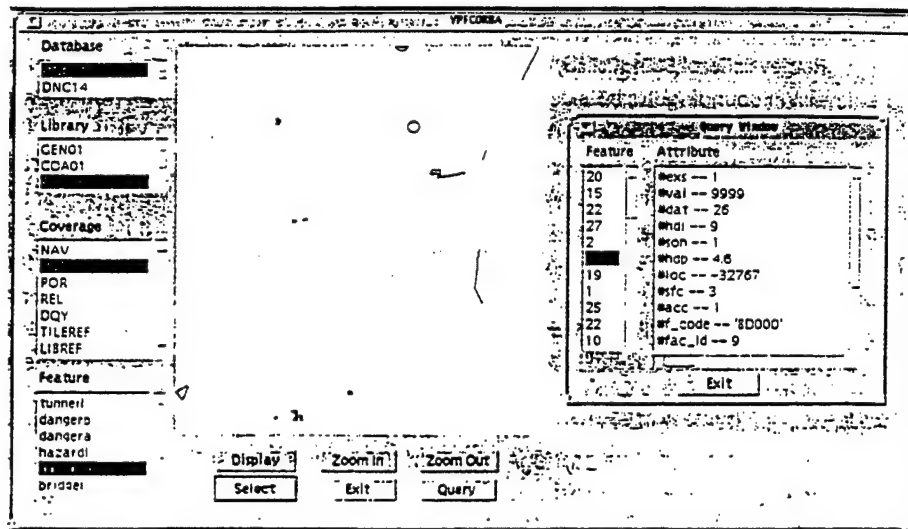


FIGURE 3. The graphical user interface for the mapping database's Java client. The main pane displays hazard areas retrieved via query; buttons below the panel allow for zooming and selecting objects and for opening a query window that lists feature IDs.

IMPLEMENTATION

We originally planned to build on the version of the Smalltalk application that had been integrated with Gemstone's commercial, object-oriented database management system. GSI was scheduled to release an ORB-integrated product in the second quarter of 1997. By the third quarter of '97, however, the product remained unavailable, so we decided to move ahead without it. We chose a two-phase migration, moving first from the memory-based application to a Corba server application, followed by a second step that integrated the changes with the ODBMS ORB when it became available. We began our work with SmalltalkBroker by DNS Technologies, primarily because GemStone planned to integrate this ORB into their ODBMS.

Our transformation of the original Smalltalk application into a server-based application focused on three areas:

- ♦ determining which objects should be server objects and thus have interfaces defined,
- ♦ inserting code to provide the Corba services needed, and
- ♦ generating the IDL from the existing Smalltalk classes.

First, we reviewed the set of classes that had direct responsibilities for feature-level management.

Because, ultimately, the server's job was to return a set of requested features to the client, we considered implementation of a single server class adequate for the application. Two classes emerged as strong candidates. VPFCoverage, a coverage-level metaclass, imports the features from the original VPF relational files and transforms them to a memory-resident OO format. VPFSpatialDataManager, a manager class, performs such functions as inserting, removing, and locating features in the quadtree spatial indexing structure. Because the relational import step that VPFCoverage handles would not be a consideration in phase two of the project (involving the ODBMS, which operates only on persistent features already stored in object format), we selected VPFSpatialDataManager to be the client's server-interface object.

We made few changes to the class's Smalltalk implementation because it already contained much of the functionality for performing the desired tasks, such as the ability to retrieve features from the quadtree contained within a specified bounding box. Our only real additions to class functionality included methods to determine and transfer appropriate database, library, and coverage names based on the client's transmitted area of interest (in the form of a VPFBoundingBox), and the issuance of messages to VPFCoverage classes to import requested data that were not located in the quadtree. We also added Corba-specific code to perform functions such as registering the VPFSpatialDataManager object

```

interface VPFSpatialDataManager : Smalltalk::Object{
    readonly attribute          any          maxLevel;
    readonly attribute          any          storedContainer;

    Retrieval::CorbaVPFFeatColl::byValue ReturnFeaturesForAOI(in
Retrieval::VPFBoundingBox::byValue aBB);
    Retrieval::CorbaStringColl::byValue ReturnDatabasesForAOI(in
Retrieval::VPFBoundingBox::byValue aBB);
    Retrieval::CorbaStringColl::byValue ReturnLibrariesForAOI(in
Retrieval::VPFBoundingBox::byValue aBB, in string dbname);
    Retrieval::CorbaStringColl::byValue ReturnCoveragesForAOI(in string
dbname, in string libname);
    Retrieval::CorbaStringColl::byValue ReturnFeatListForCov(in string
dbname, in string libname, in string covname);
    Retrieval::CorbaVPFFeatColl::byValue ReturnFeatures(in
    Retrieval::CorbaStringColl::byValue featColl, in
Retrieval::VPFBoundingBox::byValue aBB, in string dbname, in string libname, in
string covname);
    long      ReturnFeatureID();
    any       CorbaName();
};

interface CorbaVPFFeature : Smalltalk::Object{
    attribute      string      dbname;
    attribute      long      id;
    attribute      Retrieval::AttributeCollection
attributes;
    attribute      Retrieval::Coordinates      coords;
    attribute      string      covname;
    attribute      string      libname;
    attribute      Retrieval::VPFBoundingBox::byValue
boundingBox;
};

struct byValue {
    string      dbname;
    Retrieval::AttributeCollection      attributes;
    Retrieval::Coordinates      coords;
    long      id;
    string      covname;
    string      libname;
    Retrieval::VPFBoundingBox::byValue
boundingBox;
};

interface CorbaVPFFeatColl : Smalltalk::Object{
    attribute      Retrieval::FeatureCollection      featColl;

    struct byValue {
        Retrieval::FeatureCollection      featColl;
    };
};

```

FIGURE 4. The interface definition language and some significant parameter types for the VPFSpatialDataManager class. The IDL's syntax strongly resembles C++.

with the SmalltalkBroker Naming Service. Overall, these types of changes took very few lines of code to implement.

Figure 4 shows the IDL for the VPF-SpatialDataManager class, as well as some of the significant parameter types.

As Figure 4 shows, IDL's syntax strongly resembles that of C++, in which objects and their corresponding attributes and methods can be declared. As shown, the method ReturnFeaturesForAOI returns a collection of features. The byValue structures let us pass back complete copies of the features in the collection. Otherwise, the client would receive Corba object references to the objects and would have to manipulate them remotely through defined interfaces. We found this undesirable because of the client's nature as a display-and-query application. This would result in frequent system calls, which indicated that local copies of the data would be more efficient than remote message sends for acquiring pieces of feature data as needed.

Java Client

Essentially, the Corba client performs some other component's requests in the distributed application. Corba objects serve as the components that provide functionality. These objects, in turn, are specified using IDL. The system interface's specification is separate from the implementation, which lets developers make changes to the implementation without visibly altering the clients. The Corba server implements the distributed objects in the Smalltalk server, then the Java client uses them.

We developed the Corba client in Java as a standalone application using Sun's Java Development Kit. We used JavaiDL, another Sun product, for the ORB software. As the project evolved, however, we found that JavaiDL's lack of documentation hindered development. Thus, approximately one month into the development, we dropped it in favor of Visigenic Software's VisiBroker, which immediately improved client-side development time.

Because the IDL had already been gen-

erated for the Smalltalk server, we simply used the same IDL for the client, which we then compiled using the IDL compiler. The compile operation automatically generates the stub and skeleton Java code necessary to convert object references into network connections to a remote server. These references then marshal arguments of an operation to the corresponding method invocation. Project staff then wrote corresponding Java code to fill in the skeletons. Upon execution, the method returns the results to the client application. The following code segment demonstrates how an object reference is initiated and shows a subsequent method request to return CorbaVPFFeatures.

```
org.omg.Corba.Object obj =
orb.string_to_object(ior);

Retrieval.VPFSpatialDataManager
vpfRef = Retrieval.VPFSpatial
DataManagerHelper.narrow(obj);

Retrieval.CorbaVPFeature cvpfeat =
vpfRef.ReturnCorbaVPFeature
(AreaOfInterest aoi);

System.out.println("Feature name is ",
cvpfeat.dbname)
```

In the first line, the *ior* parameter is the interoperable object reference for the server object. IORs are specified by Corba as a way to provide vendor and machine-independent specifications of server objects. The IOR contains, in string format, all the information about an object that an ORB needs to locate it. The IOR is written to a file by the Smalltalk application upon creation of the server object. The Java program then reads this file and subsequently uses the IOR to initiate a server connection.

The third line of code demonstrates the actual request made to the server for the geographical features: the AOI is passed as a parameter, and the returned set of features is stored in *cvpfeat*. The user can then display and query these features locally.

Web-integrated applet

Once the standalone application worked properly, we began transforming the application to a browser-capable applet. Having the client run as a Java applet rather than a Java application required additional configurations because Web browsers impose security restrictions on Java applets' file input and output operations. To circumvent this obstacle, VisiBroker provides two utilities: GateKeeper and Smart Agent.

Essentially, the GateKeeper enables an applet to

communicate with object servers across networks without violating the security restrictions imposed by firewalls and Web browsers. The GateKeeper acts as a gateway from an applet to server objects, even if a firewall restricts access. Initiating the GateKeeper simply requires that the utility, *gatekeeper*, be initiated on the server. Doing so triggers creation of GateKeeper's IOR file, which contains important information that enables the ORB runtime inside the applet to connect with the GateKeeper.

The Smart Agent is a dynamic, distributed directory service that provides facilities for server objects and client applications. All object implementations are registered with the Smart Agent so that when a client attempts to reference an object, the Smart Agent locates the correct object implementation. Once an object is destroyed, the Smart Agent removes it from the list of available objects. Initiating the Smart Agent requires that the *osagent* utility be invoked on the server.

The resulting applet executes in the Netscape 4.0 Java-enabled browser exactly as does the standalone application, maintaining its original behavior and interface. The Netscape user is presented with the same GUI and makes requests in the same manner as described previously. The only requirement is that the server ORB continue running on the host to service incoming data requests.

Testing

At first, server and client development proceeded independently, with simple Smalltalk client and Java server code used to incrementally test the applications. This was chosen as the original development strategy because, for the most part, team members were conversant in either Java or Smalltalk, but not both. When it came time to test the Java-Smalltalk connection, however, we needed close cooperation and constant communication to fine-tune the client-server interactions.

The successful distributed application we developed proved the effectiveness of our approach and provided a foundation for a new era of research in OO digital mapping. Remaining work includes the implementation of the ODBMS/ORB-coupled server application. Another extension involves the ability to pass raster objects in addition to the vector objects already supported.

Once these tasks are accomplished, modeling and simulation users and others interested in mapping data will have access to map objects

SUPPORTING TECHNOLOGIES

In our development of the NIMA EVPF mapping database, we chose several complementary technologies to convert the application to an OO paradigm and migrate it to the Web. Chief among these were Corba, Smalltalk, and Java.

Corba

The Object Management Group developed Corba as a standard for distributed objects. The OMG itself does not develop or sell software to support the standard, but rather serves as a consortium of software vendors and end users interested in defining object standards for the industry. Commercial companies, who may or may not be members of the consortium, then develop and market products that support these standards.

As with any standard, the development of Corba has been a lengthy and arduous task. For much of the first half of the 1990s debate raged over Corba's status as a standard. Although the publications of the Corba 1.0 and 1.1 specifications seemed promising, with some vendor support being provided in com-

and marketability; thus commercial products that supported the standard, and therefore offered interoperability between vendor-specific implementations, became available. Another leap forward came with the advent of Netscape Communicator's inclusion of Visigenic Software's VisiBroker, a Java object request broker.

Corba specifies a complete middleware architecture for distributed object communication. It is one part of the Object Management Architecture published by the OMG in 1990. Specifically, Corba represents the communications element of the OMA, responsible for interoperable handling of message distribution between application-level objects. The ORB, Corba's key component, is a software bus that acts as a broker of all requests and is responsible for intercepting requests, locating the appropriate object for handling the request, and invoking the correct method on that object. The ORB must also marshal and unmarshal any parameters by converting them to a common data type and then back to a programming language-specific type, and return any results from the method invocation. Any object may be either a requestor of a service, a provider of a service, or both. Further, any two ORBs that follow the Corba 2.0 specification can provide communication between their respective application objects, regardless of vendor-specific implementation details.

The ORB communicates requests between different programming languages via the Interface Definition Language. IDL is a declarative language used to specify object interfaces and definitions. With IDL you can specify interfaces with inheritance properties, operations to which an object can respond, attributes, and types, among other things. Related IDL definitions are grouped into modules. IDL syntax itself is very similar to C++, but components defined in IDL can be implemented in any language that has Corba bindings, such as Ada, C++, Smalltalk, and Java. IDL provides the common definitions through which objects defined in different programming languages can communicate.

Another major Corba component, the interface repository, acts as a metadata repository for the ORB. The repository registers server objects' IDL interfaces for public access. Users can retrieve information about interfaces, methods, and parameters from the repository dynamically.

Finally, one of Corba 2.0's most significant contributions, the Internet Inter-ORB Protocol, enables Corba's platform and implementation independence. An IIOP message is essentially a byte stream that adheres to a network protocol. IIOP is used both in Netscape Navigator (with LiveConnect) and as the basis for remote procedure call implementation in Java.¹ An ORB must provide facilities for marshalling (converting messages into IIOP for transmission) and unmarshalling (converting IIOP packets into a local format). Figure A shows the basic Corba communication setup.

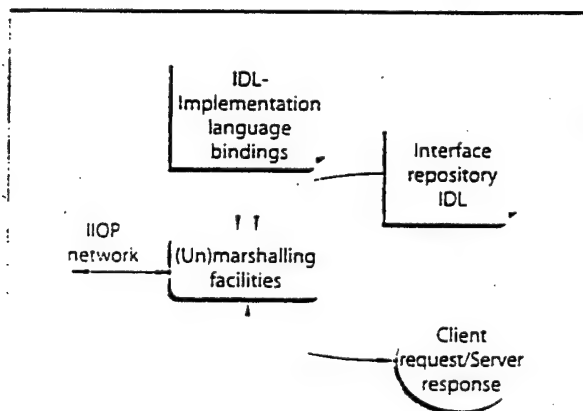


FIGURE A. Basic Corba setup. The Interface Definition Language (IDL) and interface repository facilitate communication between different programming languages, while the Internet Inter-ORB Protocol (IIOP) enables Corba's platform and implementation independence.

mercial products, there was still little or no interoperability among those products. Thus, most practitioners felt that Corba remained a work in progress.

In December 1994, OMG introduced the Corba 2.0 specification with its much-anticipated extensions to the earlier version. This version seemed to better satisfy vendors as to its stability

Jon Siegel² provides a good source for those interested in learning more about Corba, as does the specification itself.³

Smalltalk

This completely object-oriented language, introduced in the early 1980s, uses a graphical, interactive programming environment. Smalltalk is based on the concept of communicating entities, known as objects. Objects are normally regarded as instances of classes that have attributes and that implement methods for performing operations. Communication between objects occurs when one object sends a message to another, causing that object's method of the same name to be invoked. The ability to define objects that interact in this way enables the incremental development of very complex systems. Starting with just a few objects, programmers can implement basic capabilities quickly, then add to or refine these objects until the system is complete. Smalltalk's interactive environment allows changes to be made, and their effects known, in a very short period. Adele Goldberg and David Robson⁴ provide an excellent reference for those wishing to learn more about Smalltalk, and David Smith⁵ gives a quick introduction to object-oriented concepts in general.

Since the beginning of NRL's object-oriented mapping work in 1994, we have used ParcPlace-Digital's VisualWorks Smalltalk environment, in conjunction with OTI's ENVY/Developer source code manager, to let multiple developers make changes to the source code. Our development platform consists of several Sun Sparc workstations running the Solaris operating system, with one workstation per developer. This has been an extremely effective setup, and continues to be the development environment of choice for the distributed mapping project. Kevin Shaw and colleagues provide more details on the development history of the prototype and the impact of using an OO approach for the project.⁶

Java

Java's object-oriented programs, embeddable in Internet Web pages, are commonly referred to as applets. For applets to run, a reference to the actual Java program name must be spec-

ified in a Hypertext Mark-up Language document. This allows applets to be executed either within a Java-enabled Web browser such as Netscape or Internet Explorer, or by using the applet viewer provided in the Java Development Kit.

You can also use Java to create standalone applications as you would in C or C++. Although applications differ from applets in that they cannot be executed in a Web browser, converting between the two is generally straightforward. The major issue in developing applets is that such programs cannot read or write to files because of the security restrictions built into Java. You must use Corba-like software designs to overcome this obstacle.

Java's syntax is similar to that of C++. Thus, developers migrating to Java from C++ will typically encounter few difficulties. Programmers can grasp Java's fundamental concepts quickly, which helps them be productive from the outset. This proved the case for us as well. To learn more about Java, consult Ivor Horton's excellent introductory text,⁷ and James Gosling and Frank Yellin's reliable reference.⁸ We also found Andreas Vogel and Keith Duddy's Corba-specific reference⁹ invaluable for this project.

REFERENCES

1. W.R. Stanek, "Distributed Programming with CORBA and IIOP," *PC Magazine*, Sept. 23, 1997, pp. 241-243.
2. J. Siegel, *CORBA Fundamentals and Programming*, John Wiley & Sons, New York, 1996.
3. Object Management Group, *The Common Object Request Broker: Architecture and Specification*, X/Open Company Ltd., U.K., 1996.
4. A. Goldberg and D. Robson, *Smalltalk-80 the Language*, Addison Wesley Longman, Reading, Mass., 1989.
5. D.N. Smith, *Concepts of Object-Oriented Programming*, McGraw-Hill, New York, 1991.
6. K. Shaw et al., "Managing the Navy's First Object-Oriented Digital Mapping Project," *Computer*, Vol. 2, No. 9, Sept. 1996, pp. 69-74.
7. I. Horton, *Beginning Java*, Wrox Press Ltd., Birmingham, U.K., 1997.
8. J. Gosling and F. Yellin, *The Java Application Programming Interface*, Vols. 1 and 2, Addison Wesley Longman, Reading, Mass., 1996.
9. A. Vogel and K. Duddy, *Java Programming with CORBA*, John Wiley & Sons, New York, 1997.

from any platform that has a network-connected Web browser. This would let less capable client-side machines simulate the functionality of more powerful server machines. Future plans also include the ability of client users to modify data and pass the modifications back to the server for subsequent distribution. For example, in a modeling and simulation scenario, one user could perform an action that affects either the geometry or at-

tributes of a feature—for example, destroying part of a bridge—then have that information relayed back to the server so that collaborative models could be informed of the change.

After reviewing the project's first-phase implementation, we believe that the implications of migrating the original application to a server application are tremendous. Web-based mapping repositories are clearly the future for both military

and civilian mapping needs, and Corba 2.0 appears to be a step in the right direction for providing these capabilities. ❖

ACKNOWLEDGMENTS

This work was sponsored by the Defense Modeling and Simulation Office and the National Imagery and Mapping Agency's Terrain Modeling Program Office under Program Element 630603832D. We thank our TMPO program manager, Jerry Lenczowski. The views and conclusions contained in this paper are ours and should not be considered as representing those of DMSO or NIMA.

REFERENCES

1. Military Standard: Vector Product Format. Draft Document No. Mil-Std-2407. Defense Mapping Agency, Fairfax, Va., 1993.
2. M. Chung et al., "Exploitation of World Wide Web to Support Network Updating of Vector Product Format Mapping Database at a Feature Level," Tech. Report NRL/MR/7441-95-7713, Naval Research Laboratory, Stennis Space Center, Miss., 1995.
3. K. Shaw et al., "An Initial Design of Extended Vector Product Format for Modeling and Simulation," Tech. Report NRL/FR/7441-96-9651, Naval Research Laboratory, Stennis Space Center, Miss.
4. M. Abdelguerfi et al., "An Extended Vector Product Format (EVPF) Suitable for the Representation of Three-Dimensional Elevation in Terrain Databases," *Int'l J. Geographical Information Science*, Vol. 11, No. 7, 1997, pp. 649-676.

About the Authors



Maria Cobb is an assistant professor at the University of Southern Mississippi. Her primary research interests are distributed object-oriented geographical systems, the development of object-oriented techniques to improve digital mapping databases, and the application of fuzzy logic and evidential reasoning techniques to spatial data modeling.

Cobb received a PhD in computer science from Tulane University.



Harold Foley is a computer scientist with the Naval Research Laboratory and a PhD graduate student at Tulane University. His primary research interests include spatial databases, fuzzy reasoning, and developing Web-based applications.

Foley received a PhD in computer science from Tulane University.



Ruth Wilson is a mathematician for the Naval Research Laboratory. Her research interests include object-oriented programming, mathematical techniques to improve digital mapping, and distributed communication of mapping data between servers and users in real time.

Wilson received a BS in mathematics from the University of Southern Mississippi and an MS in mathematics from McNeese State University.



Miyi Chung is an electronics engineer at the Naval Research Laboratory, where she develops object-oriented techniques to improve digital mapping databases. She has also done object-oriented programming and worked with object-oriented database management systems. She is interested in the distributed communication of mapping data to database servers and tactical users in real time using Common Object Request Broker Architecture.

Chung received a BS in electrical engineering from the University of Maryland at College Park and an MS in electronics engineering from George Washington University.



Kevin Shaw is an electrical engineer with the Naval Research Laboratory, where he is the team leader for the Digital Mapping, Charting, and Geodesy Analysis Program. He has 13 years of experience in the digital mapping field.

Shaw received a BS in mathematics and an MS in computer science from the University of Southern Mississippi and an MEE in electrical engineering from Mississippi State University.

Address questions about this article to Kevin Shaw at Charting & Geodesy Branch, Naval Research Laboratory, Stennis Space Center, MS 39529; shaw@nrlssc.navy.mil.

**STUDIES IN FUZZINESS
AND SOFT COMPUTING**

**Gloria Bordogna
Gabriella Pasi**

Editors

Recent Issues on Fuzzy Databases



Physica-Verlag
A Springer-Verlag Company

Gloria Bordogna · Gabriella Pasi (Eds.)

Recent Issues on Fuzzy Databases

With 58 Figures
and 15 Tables

Physica-Verlag
A Springer-Verlag Company

Dr. Gloria Bordogna
Dr. Gabriella Pasi
Istituto per le Tecnologie Informatiche Multimediali
Consiglio Nazionale delle Ricerche
Via Ampere 56
20133 Milano
Italy
E-mail: gloria.bordogna@itim.mi.cnr.it
gabriella.pasi@itim.mi.cnr.it

ISSN 1434-9922

ISBN 3-7908-1319-2 Physica-Verlag Heidelberg New York

Cataloging-in-Publication Data applied for
Die Deutsche Bibliothek - CIP-Einheitsaufnahme
Recent issues on fuzzy databases: with 15 tables / Gloria Bordogna; Gabriella Pasi (ed.). - Heidelberg, New
York: Physica-Verl., 2000
(Studies in fuzziness and soft computing; Vol. 53)
ISBN 3-7908-1319-2

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Physica-Verlag. Violations are liable for prosecution under the German Copyright Law.

Physica-Verlag Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH
© Physica-Verlag Heidelberg 2000
Printed in Germany

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Hardcover Design: Erich Kirchner, Heidelberg

SPIN 10771506

88/2202-5 4 3 2 1 0 - Printed on acid-free paper

Uncertainty in Distributed and Interoperable Spatial Information Systems

M. Cobb¹, H. Foley², F. Petry³, K. Shaw⁴

¹ Department of Computer Science and Statistics, University of Southern Mississippi, Hattiesburg, MS 39406-5106

² Department of Computer Science, Xavier University, New Orleans, LA 70125

³ Department of Electrical Engineering and Computer Science, Tulane University, New Orleans, LA 70118

⁴ Naval Research Laboratory, Stennis Space Center, MS 39529

Abstract. Many facets of spatial data representation inherently involve issues of accuracy and uncertainty. This problem is greatly magnified when considering the integration of spatial data from different sources, such as in a distributed or interoperable environment. The general concept of schema merging involves the resolution of incompatibilities as in a distributed environment. These may be either structural or semantic in nature. Structural incompatibilities involve those, for example, in which attributes for representing the same values are defined differently. Semantic incompatibilities, however, represent those cases in which similarly defined attributes have different meanings or values. For example, an attribute of WIDTH for a road in one database may include the width of associated access lanes, while in another database it may be only the main driveable portion of the road. Such semantic issues are much more difficult to resolve, as they require a deeper understanding of the data. We will survey the issues as discussed above for spatial data in such environments and describe several approaches for different aspects of the data using fuzzy set techniques to deal with the incompatibilities.

Keywords. fuzzy GIS, conflation, uncertainty, spatial databases

1 Introduction to Spatial Data Uncertainty

The database field is currently extending its paradigm from a mostly relational approach towards an object-oriented direction. It is not clear yet what forms these object-oriented extensions will take. Indeed, there is little agreement upon which features are needed, since this approach did not spring from a single source as the relational model did from Codd's research [1]. Likewise, the scope of database capabilities is expanding to encompass multimedia, including text, pictures and sound, particularly in a distributed environment. This is of significance in supporting complex applications such as CAD/CAM design systems and geographical information systems (GIS). Such a range of database approaches represents part of a broad spectrum of information systems encompassing even the processing of large text-type files conventionally construed as the realm of information storage and retrieval systems.

The importance of representations of uncertainty in databases is increasing as more complex applications such as CAD/CAM and GIS are being undertaken in object-oriented and multi-media databases. In this paper we consider a number of issues related to uncertainty in spatial data representations and GIS, and the impact on a distributed system in dealing with them. The relational model has been the dominant database model for a considerable period, and so it was naturally used by researchers to introduce fuzzy set theory into databases. Much of the work in the area has been in extending the basic model and query languages to permit the representation and retrieval of imprecise data. A number of related issues such as functional dependencies, security, implementation considerations and others have also been investigated.

Two major approaches have been proposed for the introduction of fuzziness in the relational model. The first one uses the principle of replacing the ordinary equivalence among domain values by measures of nearness such as similarity relationships [2], proximity relationships [3], and distinguishability functions [4]. The second major effort has involved a variety of approaches that directly use possibility distributions for attribute values [5, 6, 7]. There have also been some mixed models combining these approaches [8, 9]. These approaches have also been extended to object-oriented databases [10]. We can see that variations of all of these approaches are of use in modeling spatial and geographic data, and some will be described shortly.

The need to handle imprecise and uncertain information concerning spatial data has been widely recognized in recent years (e.g., [11]), particularly in the field of GIS. GIS is a rather general term for a number of approaches to the management of cartographic and spatial information. Most definitions of a geographic information system [12, 13] describe it as an organized collection of software systems and geographic data able to represent, store and provide access for all forms of geographically referenced information. At the heart of a GIS is a spatial database. The spatial information describes the location and shape of geographic features in terms of points, lines and areas.

There has been a strong demand to provide approaches that deal with inaccuracy and uncertainty in GIS. The issue of spatial database accuracy has been viewed as critical to the successful implementation and long-term viability of GIS technology [11]. The value of a GIS as a decision-making tool is dependent on the ability of decision-makers to evaluate the reliability of the information on which their decisions are based. Users of geographic information system technology must therefore be able to assess the nature and degree of error in spatial databases, track this error through GIS operations, and estimate accuracy for both tabular and graphic output products. There is a variety of aspects of potential errors in GIS encompassed by the general term "accuracy." However, here we focus on those aspects that lend themselves to modeling by fuzzy set techniques.

1.1 Significance of Uncertainty Modeling for Spatial Data Systems

There have been a number of recent indications of the importance of uncertainty modeling in spatial data. Two in particular are of most significance. First, the

National Imagery and Mapping Agency of the United States (NIMA), announced for fiscal year 1997 a new program of University Research Initiatives. One of the major topics is uncertainty in geospatial information representation, analysis and decision support. The following are some of the main aspects:

- i. *Elements of Uncertainty*: Geospatial information is extremely complex and includes several aspects that may have associated uncertainties. These include location, relationships and typologies. They requested proposals to identify and describe all aspects of uncertainty associated with geospatial information.
- ii. *Models for Uncertainty*: The goal here was to develop extensions to existing geospatial data models that accommodate the elements of uncertainty.
- iii. *Propagation of Uncertainty*: For this aspect of the proposed efforts, they requested development of algorithms for determining how uncertainty is propagated through the fusion and analysis of geospatial information.

Secondly, the University Consortium for Geographic Information Science (UCGIS) has published a major position paper [14] of research priorities for geographic information science. In this, they state that the uncertainty information associated with a geographic data set can be conceived as a map depicting varying degrees of uncertainty associated with each of the features or phenomena represented in the data set, and potentially separable into three components: uncertainty in the typological attributes (describing the type of a geographic feature), uncertainty in the locational attributes, and uncertainty in spatial dependence (the spatial relationship with other features).

Uncertainty is seen as appearing in every part of the geographic data life cycle: data collection, data representation, data analyses, and results. The data that passes through the stages of observation to eventual archiving may be handled by a variety of individuals/organizations, each of whom may provide their own distinct interpretations to the data. So, the uncertainty is mostly a function of the relationship between the data and the user, i.e., a measure of the difference between the data and the meaning attached to the data by its current user. The UCGIS emphasized that research was needed in studying in detail the sources of uncertainty in geographic data and the specific propagation processes of this uncertainty through GIS-based data analyses, in developing techniques for reducing, quantifying, and visualizing uncertainty in geographic data, and for analyzing and predicting the propagation of this uncertainty through GIS-based data analyses.

1.2 Sources of Imprecision in Spatial Data

There is a variety of sources of imprecision in geographic information systems that are manifested as several types of uncertainty: (1) Uncertainty due to variability or error; (2) Imprecision due to vagueness; and (3) Incompleteness due to inadequate sampling frequency for missing variables [15]. Both uncertainty of interpretation and inherent ambiguity are illustrated by the labeling of data such as

that obtained from Landsat images. The images are initially processed by unsupervised classifications to obtain image classes and then the results are subjectively assigned land cover or resource class labels by a human interpreter. This is an inherently subjective task in which the interpreter attempts to match objectively derived image classes with linguistic concepts that are represented in the mind of the interpreter. It is not surprising that there is variation in the interpretation of the same data among interpreters. This is particularly troublesome when the result is stored in a database, because at this point an inherently imprecise concept requires a specific representation. As a result, uncertainty modeling in GIS has been used for the classification of land according to soil type, vegetation cover, and land use (e.g., [16]). Uncertainty representations in both probabilistic [17] and fuzzy [18] frameworks [19] have been utilized, and Fisher [20], [21] has striven to elucidate the fundamental differences between them in this context. In applications involving remotely sensed information and typical multiple sources of information used to formulate geographical data, the problems of imprecision and uncertainty are of even more concern [22].

Many operations are applied to spatial data under the assumption that features, attributes and their relationships have been specified a priori in a precise and exact manner. However, this assumption is generally not justifiable, since inexactness is almost invariably present in spatial data. Inexactness exists in the positions of features and the assignment of attribute values, and may be introduced at various stages of data compilation and database development. Moreover, inexactness may be propagated through GIS operations to appear in modified form on tabular and graphic output products. Inexactness is often inadvertent, as in the case of measurement error or imprecision in taxonomic definitions, but may also be intentional since generalization methods are frequently applied to enhance cartographic fidelity.

Models of uncertainty have been proposed for GIS information that incorporate ideas from natural language processing, the value of information concept, non-monotonic logic and fuzzy set, evidential and probability theory. For example in [23] there are reviews of four models of uncertainty based on probability theory, Shafer's theory of evidence, fuzzy set theory and non-monotonic logic. Each model is shown as appropriate for a different type of inexactness in spatial data. Inexactness is classified as arising primarily from three sources. "Randomness" may occur when an observation can assume a range of values. "Vagueness" may result from imprecision in taxonomic definitions. "Incompleteness of evidence" may occur when sampling has been applied, there are missing values, or surrogate variables have been employed.

It is now clear that uncertainty concepts must be part of the apparatus of GIS [24]. Limited positional accuracy is found in maps because of the process of map production and the inherent limitations of Earth measurement systems. Positional accuracies better than about 1 part in 10^4 or 10^5 are uncommon in paper maps [11], but much greater precision is available in computational systems. Many designers represent positions using double precision (1 part in 10^{14}), and at this accuracy, a map of the entire Earth would be capable of recording accurately the positions of large protein molecules.

Scale effects have also been considered for uncertainty modeling approaches. Maps are always generalized, to a degree that depends largely on their representative fraction or scale, or the ratio of distance on the map to distance on the ground. A map at scale 1:24,000 is less generalized than one at 1:100,000. Often, data are not available at the scale required by an application, but only at a coarser scale. As an example, to obtain a sufficiently accurate assessment of a given area's suitability for development may require data at the level of generalization corresponding to 1:24,000, but the only available digital data may be at a scale of 1:100,000. In this situation, uncertainty introduced into the results of analysis when too-coarse data are used must be considered. Ehlschlaeger et al. [25] and others have developed methods for simulating the information that is not available due to excessive generalization, based on geostatistical models calibrated in areas where both data are available at both scales.

2 Application of Fuzzy Models in GIS

2.1 Spatial Data and Attribute Modeling

Robinson [26, 27, 28] performed some of the earliest research on fuzzy data models for geographic information. He has considered several models as appropriate for this situation—the two early fuzzy database approaches using simple membership values in relations by Giardina [29] and Baldwin [30], and a similarity-based approach [2]. In modeling a situation in which both the data and relationships are imprecise, he assesses that this situation entails imprecision intrinsic to natural language which is possibilistic in nature. A possibilistic relational (PRUF) model was chosen as providing a means of facilitating approximate machine inference [31]. In the PRUF model, queries and propositions are processed by identifying constraints induced by the query or proposition, performing tests on each constraint and then aggregating the individual test results to yield an overall test score. Consider a proposition stating that a specified location is on gentle slopes and is near a certain city. The constraints induced by the proposition, "gentle" and "near," are tested using a possibility distribution yielding test results indicating the degree to which the specified location satisfies each constraint. The two test results are then aggregated to produce an overall test score indicating the degree to which the proposition is satisfied.

A number of more recent papers have also appeared that use fuzziness in modeling imprecision in spatial data. Cross [32] presents various issues in the area of fuzzy object-oriented databases and design and how these relate to topics involving GIS. Usery [33] developed an object-based model in which objects are a direct representation of the geographic entities rather than geometric elements such as point, line, and area. This design is referred to as a feature-based geographic information system (FBGIS) since the term *feature* encompasses both the geographical entity and its object representation. The approach taken is to model the spatial dimension of geographic features using fuzzy sets to define the spatial extent of the thematic dimension.

The focus is on features with undetermined boundaries which are themselves fuzzy and, regardless of the accuracy of measurement, remain fuzzy based on the concept of the feature. Examples include hills, valleys, wetlands, and many geographic features that cannot be rigorously bounded by a mathematical boundary. The model of a fuzzy feature follows that of Leung [34] using the concepts of core and boundary.

For the hill case, the core defines the prototype elevation values, i.e., all values that are unquestionably a part of the hill. The periphery is defined by the boundary and is less representative of the hill. As the complexity of the feature increases, the complexity of the fuzzy set membership function also increases. Once fuzzy set membership is defined, one can define fuzzy operations in a geographic environment. Using established fuzzy set operations such as intersection, union, and complement, a fuzzy overlay operator can be developed. Fuzzy buffer and fuzzy overlay, including intersection, union, and complement, and fuzzy boundary operators are used as have been developed by Kainz [35].

Another paper in the same volume by Sarjakoski [36] emphasized issues related to identity of geographical objects. This issue arises in the practical context of identifying objects in a complex and dynamic environment; for example, the question of enumerating how many lakes, islands and rivers are in Finland.

Existence is a necessary property of an object. An object also has an identity. This is the key issue when distinguishing between field- (in the mathematical sense) and object-oriented approaches in describing geographic reality [37, 38]. There is no identity related to fields as such. Objects can be, and very often are, associated with regions in a field; thus, fields are indirectly connected with the concept of identity.

Many of the above concepts are closely interwoven. Take the triple existence-identity, for example. It is difficult, or even impossible, to speak of one without another. According to our contemporary understanding of the universe, geographic objects can only exist confined in four-dimensional space-time. Where there appears to be fuzziness, it occurs in many of the properties of an object, not in one alone. Some of the extreme cases are:

Fuzziness only in 3D space: Fuzziness may be limited to the spatial extent of a geographical object, i.e., the boundary of the object is ill-defined, even though there is no doubt about its identity and temporal extent.

Fuzziness only in time: The temporal extent of a geographic object may be ill-defined, even though the spatial extent is well-defined, i.e., the spatial boundary is sharp.

Fuzziness in identity: This is closely related to fuzziness in spatial and temporal extent. The issue here is whether an object has enough identity to be considered individual. With dynamic objects (objects that change in time), the concept of identity is linked to the problem of the extent to which an object can change and still be the same object.

Fuzziness in class definition: Class definitions are sometimes fuzzy and overlap, making it impossible to assign an object to only one class.

Fuzziness in class membership: Even when there are well-defined (axiom- or definition-based) classes, the characteristics of an object may still be such that we cannot be quite sure to which class it belongs.

In an actual survey, heuristic and rather informal classification rules were used to define lakes, rivers, and islands. Much criteria cited involved linguistic descriptions and modifiers.

In principle and by definition, a geographic object will always have an identity and therefore a crisp existence. This implies that a geographic object can always have an individual name. Sarjakoski's case study demonstrates that, in reality, it is often very difficult to state whether or not something can be modeled as an individual object. The objects on this conceptual borderline can be said to be fuzzy in existence. In other words, it is not clear whether the object exists at a fuzzy existence is always strongly related to fuzziness in extent, whether spatial or temporal. The task of counting the number of geographic objects deals only with their existence; it is in principle unnecessary to know their extent. Nevertheless, because the objects are located in the same geographic field, their spatial extent must be determined in a consistent way.

A recent system developed by Robinson is the Knowledge Based Land Information Manager and Simulator (KBLIMS) [39] which can provide a perspective on fuzzy sets as a basis for managing certain kinds of uncertainty in a geographic information system.

Several domains were considered for the use of fuzzy sets to model uncertainty in a comprehensive, intelligent geographical information system application such as KBLIMS. On the terrain analysis level, uncertainty representation is required for the hydro ecological simulation system as well as the intelligent query portion of the system. The uncertainty model of Davis and Keller [40] shows how to incorporate fuzzy metadata concepts with Monte Carlo simulation in a system that can use the uncertainty information to provide an assessment of the validity of the model output. The most direct approach for the integration of fuzzy sets at the terrain level in KBLIMS has the uncertainty within the process of terrain analysis so that the output is in a crisp form. Otherwise, the objects formed from the terrain analysis would need a capability to represent and interpret uncertainty at the object model level.

The object model level of KBLIMS is where management of uncertainty using fuzzy sets can be made most explicit and thoroughly integrated into the system. Incorporation of fuzzy instances, objects, and classes depends greatly upon the nature of the data output from the terrain analysis and the needs/capabilities of the query model. In this way, the traditionally separate fields of representation and spatial analysis [41] become integrated. A working object model in KBLIMS requires implementation of the behavior of the fuzzy objects, relations, or classes. Often, this would be the same as many spatial analysis functions, except that some spatial analysis functions would be incorporated into portions of the query model.

KBLIMS and other sophisticated GIS applications have not been initially designed with the use of fuzzy sets in mind for uncertainty management. There are portions of the system which may or may not use fuzzy sets to address very specific problems such as model self-evaluation, allowing the system to enhance its ability to manage uncertainty generated by its own activities. However, as the application domain evolves and experience with formalizing that domain increases, some significant kinds of uncertainty may be specified and ultimately managed with fuzzy sets. It is also evident that in GIS applications such as KBLIMS, that simply arriving at membership values to describe variations in soils and their properties over space is only the beginning of the development of a process for formalizing and managing uncertainty that will spread through the system, requiring changes at the object model level.

2.2 Spatial Relationship Modeling

The manner in which spatial data is modeled affects important aspects of its use, including querying capabilities and relationship inferences. Especially important for spatial data is the ability to model relationships between spatial features. Such relationships can provide a significant resource for processing spatial queries.

A spatial data model that provides a representation for storing information concerning binary relationships between two-dimensional objects is described in previous work [42, 43, 44]. The binary relationships incorporated include both qualitative topological and directional relationships. In this approach, "qualitative topological relationships" include both those relationships based on topological invariants, e.g., Egenhofer [45], and a more intuitive, less formal set of relationships based on various other properties. The model represents a compromise between storage and performance issues by explicitly storing information that can easily be used for inference purposes by a fuzzy query framework, such as that given in [44, 46].

Minimum Bounding Rectangles (MBRs) are used as the basis for object representation. This approach, as well as that of Nabil [47] Sharma [48] and Clementini [49], relies upon the use of MBRs as approximations of the geometry of spatial objects. The use of MBRs in geographic databases is widely practiced as an efficient way of locating and accessing objects in space [49]. In addition, numerous spatial data structures and indexing techniques have been developed that exploit the computationally efficient representation of spatial objects through the use of MBRs [50, 51].

This model also supports fuzzy relationship representation and querying of specific aspects of direction and qualitative topology. For direction, it is used to answer queries related to the *degree* to which one object lies in a particular direction with respect to a second object. The determination of the degree is made through calculations that utilize area and axis ratios of the objects involved. The resulting quantitative value is then mapped to a range that corresponds to a term known as a *directional qualifier*. Directional qualifiers can be used in queries to

determine, for example, that *object A is mostly west of object B*, or that *object A is somewhat southeast of object B*.

The need for such directional qualifiers is evident when considering the difficulty of determining a precise directional relationship for 2-D objects. The areal extent of 2-D objects often results in the possible application of more than one directional relationship. Often, centroids are used to provide a single point of reference for determining orientation of 2-D objects, (e.g., Chang [52]). However, some information loss is inherent in the centroid method. This model preserves all of the directional relationships that exist between two 2-D objects, along with a representation indicating the degree to which the objects are analyzed to participate in each of the relationships. This method of qualifying directional relationships more closely models the way in which humans naturally process and communicate such information.

Qualitative topological relationships such as *overlaps*, *surrounded-by*, *disjoint*, etc., are similarly modeled so that the degree to which each of the objects participates in the relationship is stored for subsequent fuzzy querying. Determination of this information utilizes various combinations of area ratios, ranges of which are mapped to a set of linguistic qualifiers such as *some*, *most*, *all*, etc., in a manner analogous to that used for directional relationships. These qualifiers allow users to distinguish the various cases of the same relationship. For example, given that A overlaps B, does *all* of A overlap *some* of B, or does *little* of A overlap *most* of B? This type of distinction is frequently made during human-to-human interaction, but little support for it has been incorporated into computer data models.

The definition of the relationships is based on an extension of Allen's temporal relations [53] to the spatial domain. This is done by allowing each of Allen's thirteen relations to represent the interaction of two-dimensional objects in terms of an x and y relationship component. The resulting set of relationships is then used for defining topological and directional relationship terminology. A data structure called an abstract spatial graph (ASG) is defined for the binary relationships. The ASG maintains all necessary information regarding topology and direction, and provides the basis for processing of fuzzy topological and directional queries.

The ASG model was specifically designed to store explicit information regarding fuzzy topological and directional relationship information for pairs of objects. This correlates well with the observation in [45] that topological information should be considered first-class information within the realm of spatial databases.

Briefly, an ASG is a directional polar graph with three types of nodes: (1) nodes representing portions, or object sub-groups of the first object participating in the relationship, (2) nodes representing object sub-groups of the second object, and (3) nodes representing overlapping areas and/or reference areas. For the graphical representation of an ASG, the various types of nodes are differentiated through color and symbol style. As stated earlier, one of the basic assumptions for the data model and subsequent query framework is that objects, or features, are enclosed within MBRs. Reference areas for relationship calculations are defined

as being either one of the object sub-groups, a specially defined rectangular area, or a line segment--depending upon the general categorization of the relationship involved.

Each node of an ASG has a pair of associated weights: an area weight and a total node weight, that provides information to support fuzzy querying. Specifically, these weights are used to define fuzzy qualifiers that answer queries such as, "To what degree is area A east of area B?" or, "How much of area A surrounds area B?." Total node weights support queries of the first type, while area weights support queries of the second type. Ranges of weights are used to define qualitative terms useful for describing these concepts, such as "most," "some," "slightly," or "mostly." The manner in which the weights are computed is described in [43].

The ASG data model supplies the infrastructure upon which a fuzzy query system is built. This system, topological query framework (TQF), includes a grammar in Backus-Naur Form (BNF), data structures and query processing strategies. Fuzzy relationship querying follows naturally from the ASG model due to its support for direct binary relationship representation and weighting system. Previous work [46] has shown how TQF can be integrated with several existing spatial query languages to provide fuzzy querying extensions.

3 Conflation

Conflation is typically regarded as the combination of information from two digital maps to produce a third map that is better than either of its component sources. The history of map conflation goes back to the early to mid-1980's. The first clear development and application of an automated conflation process occurred during a joint United States Geological Survey (USGS)-Bureau of the Census project designed to consolidate the agencies' respective digital map files of U.S. metropolitan areas [54]. The implementation of a computerized system for this task provided an essential foundation for much of the theory and many of the techniques used today. Since that time, others, including commercial GIS vendors, have implemented conflation tools within their applications. For an example of commercial work on conflation, see [55].

Conflation of maps typically is needed because: (1) users wish to update their mapping information without losing legacy data which may not be included in the new information; (2) one map source may be more accurate with respect to, e.g., positional or attribute information; or (3) one map source contains information missing in another, such as additional features, feature attributes or even entire coverages. Motivation for this work includes all three of these reasons and represents an effort to more fully utilize associated non-spatial data as well as spatial properties in an "intelligent" type of system that more closely mirrors the way that human experts would manually conflate mapping data.

Conflation can, in brief, be viewed as a multi-step, iterative process that involves feature matching, positional re-alignment of component maps and attribute deconfliction of positively identified feature matches. Feature matching, simply and perhaps somewhat obviously stated, involves the identification of

features from different maps as being representations of the same geographic entity. Positional alignment is a mathematical procedure in which previously identified matching features are brought into spatial agreement, while deconfliction is a process in which contradictions in a matching pair's attributes and/or values are resolved. Positional alignment and deconfliction are both steps that are performed after a positive match has been determined. As such, it is easy to see that accurate feature matching results are essential to the overall quality of the resulting conflated map. Because of this dependency, our work thus far has concentrated solely on feature matching aspects of conflation. Our motivation for this work includes an effort to make individual geographic features "intelligent" enough to know when and how to conflate themselves in a distributed environment.

3.1 Conflation and Uncertainty

The assessment of feature match criteria is a process in which evidence must be evaluated and weighed and a conclusion drawn—not one in which equivalence can be unambiguously determined. For example, fuzzy concepts such as "closeness" of two features and "similarity" of attributes and feature groupings are essential for determining equivalence.

In fact, feature matching can be considered as a type of classification problem. That is, we are trying to determine whether one feature belongs to the same "class" as another; in this case the class consists of members believed to be equivalent. This type of problem can be handled through theories of evidential reasoning or uncertainty, such as fuzzy logic [57] or Dempster-Shafer theory [58]. These theories attempt to provide likelihood measures for questions based on available, though not necessarily conclusive, evidence. For example, in feature matching, the question we must consider is, "Based on the available evidence, what is the likelihood (or probability) that feature A from map coverage 1 represents the same entity as feature B from map coverage 2?"

Dempster-Shafer theory is a method of inexact reasoning based on the modeling of uncertainty as a range of probabilities. It provides representations for the degree of belief in evidence (belief that supports a hypothesis), as well as the degree of disbelief in evidence (belief that refutes a hypothesis) and the nonbelief (neither belief nor disbelief). That is, it not only supplies results regarding the belief that two features are the same, but it also provides a representation of how much it is disbelieved that they are the same, along with the degree to which there is no evidence either way.

The approach described here for feature matching draws from aspects of both fuzzy set logic and evidential reasoning. In particular, the assignment of matching scores for linguistic attributes is directly motivated by work in modeling linguistic variables by the use of fuzzy sets. For example, we need to be able to determine the semantic similarity of an attribute such as **road surface type** which may have a value of 'hard' for one feature and 'asphalt' for the potential matching feature. Obviously, the semantics of the two are not strictly equivalent, but neither are they

contradictory. Likewise, the idea of combining scores from the different components of feature matching to arrive at a single matching score is very similar to techniques for the combination of evidence used in evidential reasoning.

Previous work [59, 60, 61] has concentrated on developing an uncertainty model for feature matching. Human cartographic experts are skilled in making mapping decisions based on conflicting information, and it is this skill at reasoning with uncertain information that we attempted to mimic in this system. Geographic features represented in different physical databases are very unlikely to have completely identical information due to differences in collection instruments, human interpretation, processing techniques and varying representations of the data. Therefore, a conflation model must be able to determine when such differences are truly indicative of non-matching features, and when such differences could be attributed to one of the causes identified above.

The technique developed in [59] is able to accommodate cases where values for corresponding attributes differ, as well as cases where the attribute sets themselves differ. For implementation, each feature is considered as a set of attribute-value pairs:

$((a_{11}, v_{11}), (a_{12}, v_{12}), \dots, (a_{1n}, v_{1n}))$
 $((a_{21}, v_{21}), (a_{22}, v_{22}), \dots, (a_{2m}, v_{2m}))$
 \dots

From this representation, a degree of matching similarity is determined. A different approach is used for different attribute value domains. For numeric domains, a membership matching function is used, while a similarity table is used for linguistic domains. Our approach to linguistic attribute matching is to establish a similarity value s (in the range $[0, 1]$) for each pair of elements of the domain. This value is determined from the semantics of the domain and the linguistic terms. The characteristics of the similarity function s are:

$$s_A(x, y) = s_A(y, x) \quad \text{symmetric}$$

for all values $x, y \in \text{domain of attribute A}$. For well-defined values of the domain (e.g. not "unknown" or "other")

$$s_A(x, x) = 1 \quad \text{reflexive}$$

where x is a well-defined value.

As an example, we consider the Railroad attribute *RRA* (Railroad Power Source), which is restricted to the values (0—Unknown, 1—Electrified track, 3—Overhead electrified, 4—Non-electrified, 999—Other). The similarity table for *RRA* is shown in table 1. Since linguistic similarity is symmetric, we need only show the lower triangular values in the table. Note that since 1, 3, and 4 are well-defined linguistic terms, they are shown with the reflexive value of 1 on the diagonal, e.g. $s_{RRA}(3, 3) = 1$. However, since 0 and 999 are non-specific categorical values for this particular domain, their diagonal values were

determined to be less than 1, reflecting the potential lack of exact matching for such linguistic terms.

Table 1. Similarity table for attribute *RRA*.

RRA	0	1	3	4	999
0	.2				
1	.2	1			
3	.2	.6	1		
4	.2	.1	.1	1	
999	.2	.2	.2	.2	.2

Because most features have more than one attribute, we must also consider semantic interrelationships among the attributes in determining matching features. These are represented as rules in the expert system which return associated weights. These weights are used either to add credence to the hypothesis of matching features, or to weaken it. As an example, consider the Railroad attributes *LTN* (Number of tracks) and *RRC* (Railroad category). The rule for the relationship between the two attributes is expressed as:

IF ((RR1.ltn = 3 and RR2.ltn = 2) and
 (RR1.rrc = 16 and RR2.rrc = 16))
 THEN $w_m \leftarrow 1.0$
 $w_{ltn} \leftarrow 0.5$,

where RR1 represents the first Railroad object and RR2 represents the second. This rule illustrates a conflict in the values of the length attribute of the two features. We see from the example that the resulting weight for *LTN* is weakened, giving it less influence than that of *RRA* in the combined matching score.

A composite matching score is then computed from the combination of the expert system weights and the similarity table values. This score is given as:

$$MS_{i,j} = \left(\sum_{k=1}^N [simA_k(F_i, F_j) \times ESW_{A_k}] \right) / N$$

where

A_k = k th attribute in both F_i and F_j , where $0 \leq k \leq N$.

N = number of attributes that describe both F_i and F_j .

ESW_{A_k} = weight associated with A_k computed by the expert system.

3.2 Geometric Analysis

The second part of this work consists of a technique for qualitatively matching linear features based on shape similarity. This broadens the basis for feature matching by adding a spatial component to the non-spatial attribute-matching algorithm presented above.

The first step involves a pre-processing of the features to bring them into a standard position for the shape similarity analysis. Translation, rotation, and scaling transformations are applied in sequence to accomplish this positioning. Translation is performed by moving the features' starting nodes to the origin of planar axes. Then, the end nodes are rotated to the x-axis. Finally, the features are scaled so that the starting and ending nodes are equal to [0.0] and [1.0], respectively. These transformations allow us to more reliably compare linear features of different sizes. Figure 1 shows the results of standardizing two linear features through this process.

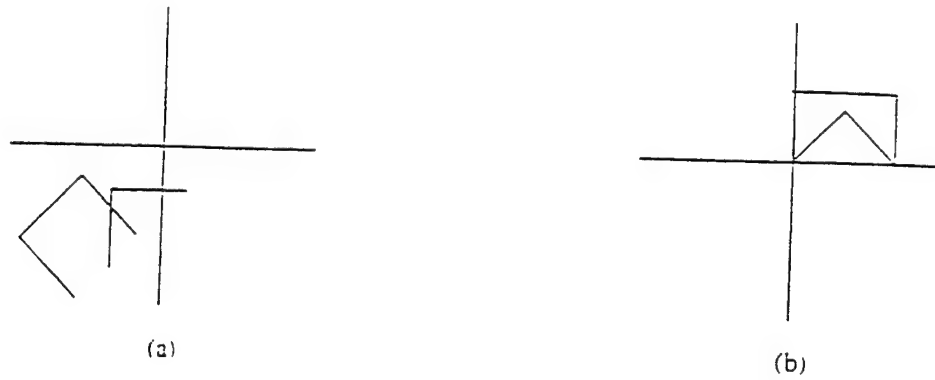


Figure 1. Linear features (a) before and (b) after standardization process.

After the features have been standardized, a process called *node merging* takes place [54]. This requires all nodes from one feature to be mapped onto the other feature, and vice versa. This is performed on the basis of a node's distance from the starting node of the feature. Consider two features, F1 and F2, comprised of nodes $(n_{11}, n_{12}, \dots, n_{1j})$ and $(n_{21}, n_{22}, \dots, n_{2k})$, respectively, and where the associated ratios are given as $(r_{11}, r_{12}, \dots, r_{1j})$ and $(r_{21}, r_{22}, \dots, r_{2k})$ such that $r_{11} = r_{21} = 0.0$ and $r_{1j} = r_{2k} = 1.0$, where r_i is the ratio of the distance from start to node n_i to the overall length of the linear feature. Then, new features F1 and F2 are created that each have the same number of nodes such that each node is the result of merging the features' nodes based upon their ratios.

Given that the features are in standard position and the nodes have been merged, we begin the last phase in assessing shape similarity. This involves determining a normalized distance measure between corresponding nodes of the two features, given as

$$SHsim = \left(\sum_{i=1}^{j+k-2} Dist(n_{1i}, n_{2i}) / (D_1 + D_2) \right)$$

where $j + k - 2$ is the number of distances to be computed, and D_i is the total length of F_i . This implies that features with a higher shape similarity measure are more likely candidates for matching features. It is based on Frechet's measure of distance, L_2 - Distance, between polygonal arcs [55]. Suppose that we have two functions

$$\begin{aligned} P &: [0,1] \rightarrow \mathbb{R}^2 \\ Q &: [0,1] \rightarrow \mathbb{R}^2 \end{aligned}$$

where (p_0, p_1, \dots, p_n) and (q_0, q_1, \dots, q_m) are ordered point sets for P and Q , respectively. Then the set of ratios used for computing the distance between P and Q at all of the r_i 's is computed as described for node merging above. Thus, for Frechet's measure, the following is used.

$$\begin{aligned} L_2 &= \left(\int_0^1 \|P(r) - Q(r)\|^2 dr \right)^{1/2} \\ &= \left(\sum_{i=1}^{n+m} \int_{r_{i-1}}^{r_i} \|P(r) - Q(r)\|^2 dr \right)^{1/2} \end{aligned}$$

Now, given that (x_{P_i}, y_{P_i}) and (x_{Q_i}, y_{Q_i}) are the coordinates of $P(r_i)$ and $Q(r_i)$, respectively, and $\Delta_{x,i} = x_{P_i} - x_{Q_i}$ and $\Delta_{y,i} = y_{P_i} - y_{Q_i}$ are the measured separations of the linear features at the bending points of either of the lines, then the closed form of the equation for L_2 - Distance is represented by

$$\left\{ \frac{1}{3} \sum_{i=1}^{n+m} \left((r_i - r_{i-1}) \left(\Delta_{x,i-1}^2 + \Delta_{x,i-1} \Delta_{x,i} + \Delta_{x,i}^2 + \Delta_{y,i-1}^2 + \Delta_{y,i-1} \Delta_{y,i} + \Delta_{y,i}^2 \right) \right) \right\}^{1/2}$$

Of course, the mathematical exposition presented here presents the objective portion of feature matching. The subjective portion represented by the expert system is much more difficult to capture. In response to this problem, Foley et al. [60, 61] present a web-based knowledge acquisition tool for capturing expert knowledge needed for conflation.

4 Distributed Spatial Systems

In recent years, the trend for most types of information systems has been a move to a more loosely coupled, distributed nature. The maturity of client-server architectures and software, as well as the virtual explosion of web-based systems, has demonstrated the enormous advantages of distributed systems. Furthermore, the advent of successful middleware technology such as the CORBA 2.0 standard and corresponding vendor implementations has drastically reduced barriers to communication and data sharing among heterogeneous software and hardware systems.

The interest in distributed GIS is no less than that of general information systems; however, the uniqueness of the nature of spatial data makes the issue of true *interoperability* of GIS a major research concern. Evidence of this is abundant in the literature, and is also illustrated by initiatives such as the Open Geodata Interoperability Specification (OGIS) work by the Open GIS Consortium, Inc., as well as UCGIS priority research panels [14] on "Interoperability of Geographic Information" and "Spatial Data Acquisition and Integration."

4.1 Uncertainty and Conflation for Distributed Data

Obviously, issues of uncertainty that apply to conflation within a single system—such as those for feature matching—are also applicable to conflation in a distributed environment. However, we believe additional factors related to the general topic of distributed databases increase the scope of uncertainty that must be considered in this context. As background, we can draw from the abundance of past and ongoing research in the realm of schema merging for conventional (i.e., relational) distributed heterogeneous databases. An example of work in this area includes [62].

The general concept of schema merging involves resolution of incompatibilities in metadata. These incompatibilities may be either *structural* or *semantic* in nature. Structural incompatibilities involve those, for example, in which attributes for representing the same values are defined differently. These may include different names for the attributes, or different domains for their associated values, e.g., float vs. integer. Semantic incompatibilities, on the other hand, represent those cases in which similarly defined attributes have different meanings or values. For example, an attribute of width for a road in one database may include the width of the road plus any associated right-of-ways, while the same attribute name in another database may only imply the width of the paved/driveable portion of the road. Semantic incompatibilities are much more difficult to handle automatically, as they necessarily imply a deeper understanding of the data.

It is clear that these issues are very similar to ones that must be faced in performing conflation in distributed spatial databases. In particular, semantic schema integration and the feature-matching phase of conflation require similar levels of knowledge regarding the meanings that various data are intended to convey. Semantic knowledge is inherently uncertain, as interpretations of even the most seemingly unambiguous words and phrases vary among individuals. Similarly, structural differences in spatial data representation for like features are to be expected in any distributed system comprised of heterogeneous data sources. It is evident from this discussion that conflation in a distributed environment can be viewed as a specific application of issues related to uncertainty in schema merging.

In keeping with generally accepted principles of distributed (as well as non-distributed) database design, the model presented in [63] is completely independent of physical concerns such as data partitioning. However, the general issue of considering conflation within a distributed system versus a standalone

system does impact design decisions, even at an abstract logical level. The primary issue considered in this work is centralization versus distributed control of conflation events.

4.2 Object-Oriented Modeling Approach

The logical design for this approach is based on an object-oriented (OO) model. The OO paradigm is well accepted as the prevailing method for the representation and manipulation of complex data such as geographical information. Within an OO framework, one is able to define models of real-world data in ways analogous to those in which we intuitively perceive and interpret those data. As a general introduction to the subject, an *object* is a collection of data (state) and methods (behavior) that represents the properties and processes of a real-world entity, such as the Chesapeake Bay Bridge or the Mississippi River. A *class* is a template for creating new objects that share common properties. For example, there may be a bridge class that captures generic information that every *instance* of that class (an instantiation) should contain. This would most likely include data such as length, height, maximum weight limit, and type (drawbridge, suspension, etc.). Examples of procedures for a bridge class could include opening and closing.

The packaging of an object's data with its procedures is known as *encapsulation*. Encapsulation allows modifications/additions to the system with minimal impact on other system components. This property is crucial for the successful development and maintenance of complex software systems such as GIS. Other major concepts for understanding OO models include *inheritance* and *polymorphism*. Inheritance is the automatic inclusion of attributes and methods from classes defined at a higher level in a class hierarchy. The class hierarchy is designed with more general classes being placed at higher levels and more specific classes being placed at lower levels. Inheritance reduces the need to duplicate data and code, while the structuring of a class hierarchy provides a logical organization of object "types."

Polymorphism allows methods for different objects to have the same name, while providing different implementations. For example, both a circle and a square class could implement a method that calculates area. Both methods could be named *area*, but the implementations would use the appropriate formula for either a circle or a square. Methods that invoked the *area* method for an object would use one name--the class of the receiving object would determine which implementation to use. Polymorphism helps to simplify system design, and reduces coding complexity by eliminating error-prone if-then-else type-checking constructs.

These concepts are applied to our work in the following ways. Encapsulation allows each geographic feature to maintain its own state of knowledge related to information and procedures necessary for conflation with another feature. Changes in the algorithms/implementations of a particular feature's conflation abilities do not affect other features. Inheritance allows us to describe general types of conflation knowledge applicable to multiple feature classes within a single class at a high level in the hierarchy. Lower-level classes that are more

specific then automatically inherit this knowledge. Finally, polymorphism allows us to implement general conflation procedures that are valid for instances of any feature class; thus, polymorphism greatly reduces the need to be concerned with the issue of "type." The result is that we are able to treat, for example, railroad features and building features in the same manner at a logical level. Figure 2 shows a simplified class hierarchy for conflation.

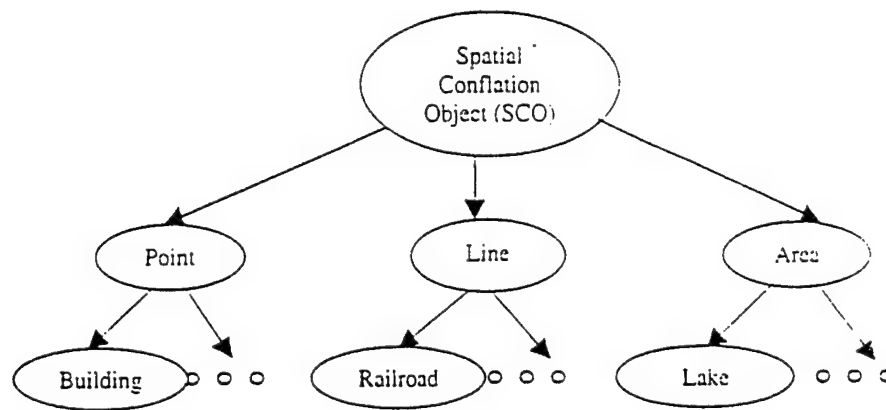


Figure 2. General class hierarchy for conflation.

The Naval Research Laboratory's Geospatial Information Database (GIDB) prototype [64], within which proof-of-concept implementation of this model is currently being performed, is centered on the concept of spatial range queries, also known as area-of-interest (AOI) queries. Given an AOI, through definition of manual or graphical bounding box coordinates or geographical place name, the GIDB is able to return any vector, raster or multimedia data available for that area. Advanced queries, such as those limiting vector data attribute values, are also available.

Our conceptual model of the conflation process is based on this system model of AOI queries, limited to the consideration of vector data only. A diagram of the conflation process is shown in figure 3. The primary point to note is that the process takes place at the individual feature level, irrespective of that feature's physical residence, or logical database, library or coverage inclusion. In the first step, the user selects an AOI. The query manager then retrieves all feature objects from the distributed database that fall within the AOI (subject to any other constraints imposed by the user). From this collection of objects, the query manager randomly selects one object to send a "conflate" message. That object follows the protocol for determining which, if any, of the remaining objects in the query collection are matching representations for itself. Any object determined to be a match is placed in the matching feature set for the conflating object, ranked

according to similarity scores. Objects that have been placed in a matching feature set are removed from the general query collection, so as not to be candidates for subsequent conflation iterations. This philosophy implies that only those objects that have scores *strongly* suggesting matching features are placed in the matching feature sets. The query manager continues by sending "conflate" messages to the remaining members of the query collection. When the process is complete, the query manager returns the query collection for display. For any features with non-empty matching feature sets, the member of the set with the highest quality score (NOT the same as the similarity score) is returned as the representative for that feature. The quality score is an indication of the fitness for use of the data, and is based on multiple factors derived from both the data and metadata.

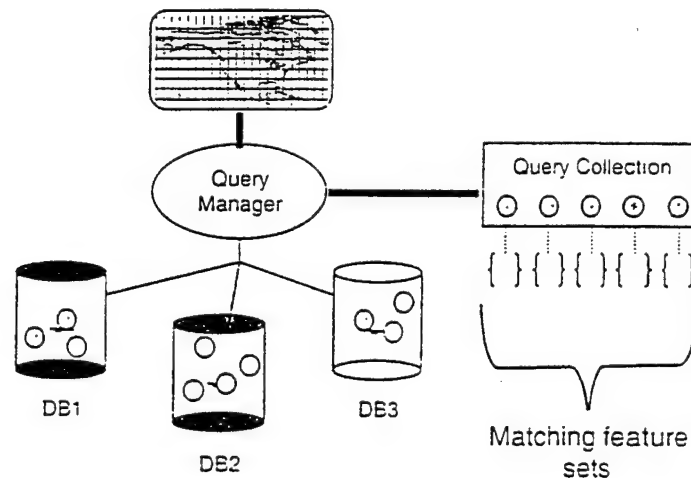


Figure 3. Conceptual model for distributed conflation.

Several points on the conceptual model are worth noting at this point. First, the responsibility for conflation is distributed. Each individual feature contains the inherited knowledge needed for performing feature matching for that particular class of geographic objects. This is preferable to a centralized conflation system, where a single conflation object "manages" the process. Such a system would be more difficult both to implement and maintain due to differences in the ways in which objects from various feature classes should be matched, as well as the ramifications of adding new features and/or system nodes for a distributed system. Second, the process is automatic. Because the query manager collects the features satisfying the user query and invokes the conflation method for each object before returning the final set, the user does not have to explicitly request conflation, or even need to know that conflation is an issue. Third, when the results are returned to the user, only a single representation of each object is presented. For now, our treatment of deconflation is simply to select the "best" feature from a matching

feature set, based on various objective and subjective criteria. This idea of simplifying the user's concern and involvement with conflation is critical for end-user based systems, as the whole need for such an automated approach is derived from the concept of non-expert users. A more detailed explanation of the model can be found in [63].

Though irrelevant at a conceptual level, physical allocation of data in a distributed system is of paramount concern for performance issues. Here, we briefly consider one possible allocation scheme and the potential impact as related to the model.

The scheme we consider is a partitioning based on representational classes. That is, all line features such as railroad lines, power lines, etc., reside on the same physical node; all point features are likewise allocated to the same node, as are all area features. In consideration of the hierarchy given in figure 2 within this setup, we believe the optimal partitioning of the hierarchy is to include all level 3 (feature class) classes and their instantiations together with their corresponding level 2 superclass. For example, all transportation lines, utility lines, etc. would be co-resident with the line class object. For conflation, this setup would minimize network traffic needed for transfer of data and execution of methods, as most of the specific conflation knowledge is inherited from the SCO and values instantiated at these lower two levels. Of course, other partitionings may also provide acceptable performance, but this one is given as an example of considerations between data allocation methods and the distributed conflation model.

5 Concluding Remarks

The importance of uncertainty modeling for spatial data and GIS is well recognized in the geographic information science research community. We have surveyed the relevant issues and some specific efforts at spatial modeling using fuzzy set approaches. As distributed systems and web-based interoperability have come to the forefront, uncertainty issues are then reflected in such an environment.

Our current research directions are to more fully develop formal approaches to distributed spatial data semantics and to produce prototypes based on these approaches.

Acknowledgments

This work was sponsored by the Marine Corps Warfighting Lab (MCWL) under Program Element 63640M, with LtCol David Durham, LtCol Jerry Brown, and MAJ Mike Berigan serving as program managers.

References

1. E. Codd, "A Relational Model for Large Shared Data Banks," *Communications of the ACM*, Vol.13, pp.377-387, 1970.

2. B. Buckles and F. Petry, "A Fuzzy Model for Relational Databases," *Int. Jour. Fuzzy Sets and Systems*, Vol. 7, pp. 213-226, 1982.
3. S. Shenoï and A. Melton, "Proximity Relations in Fuzzy Relational Databases," *Int. Jour. Fuzzy Sets and Systems*, Vol. 31, pp. 287-296, 1989.
4. M. Anvari and G. Rose, "Fuzzy Relational Databases," *The Analysis of Fuzzy Information Vol. II*, (ed. J. Bezdek), pp. 203-212, CRC Press, Boca Raton, FL, 1987.
5. M. Umamo, "FREEDOM-0 : A Fuzzy Database System," *Fuzzy Information and Decision Processes*, (eds. M. Gupta and E. Sanchez), North-Holland, Amsterdam, pp. 339-347, 1982.
6. H. Prade and C. Testemale, "Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries," *Information Sciences*, Vol. 34, pp. 115-143, 1984.
7. M. Zemankova and A. Kandel, "Implementing Imprecision in Information Systems," *Information Sciences*, Vol. 37, pp. 107-141, 1985.
8. E. Rundensteiner, L. Hawkes, and W. Bandler, "On Nearness Measures in Fuzzy Relational Data Models," *Int. Jour. Approximate Reasoning*, Vol. 3, pp. 267-298, 1989.
9. J. Medina, O. Pons, and M. Vila, "Gefred: A Generalized Model to Implement Fuzzy Relational Databases," *Information Sciences*, Vol. 47, pp. 234-254, 1994.
10. R. de Caluwe, ed., *Fuzzy and Uncertain Object-Oriented Databases*, World Scientific, Singapore, 1997.
11. M. Goodchild and S. Gopal, eds. *The Accuracy of Spatial Databases*, Taylor and Francis, Basingstoke, UK, 1990.
12. —, *ARC/INFO User's Guide: ARC/INFO 6.0 Data Model: Concepts and Key Terms*, Environmental Systems Research Institute, Redlands, CA.
13. D. MaGuire, "An Overview and Definition of GIS," *Geographical Information Systems: Principles and Applications, VOL 1 - Principles*, (eds. D. MaGuire, M. Goodchild, and D. Rhind), pp. 9-20, Longman, Essex GB, 1991.
14. University Consortium on Geographic Information Science, "Research Priorities for Geographic Information Science," *Cartography and Geographic Information Systems*, Vol. 23, No. 3, pp 115-127, 1996.
15. H. Veregin, "A Taxonomy of Error in Spatial Databases," Technical Report 89-12, National Center for Geographic Information and Analysis, Santa Barbara, CA, 1989.
16. P.A. Burrough, "Fuzzy Mathematical Models for Soil Survey and Land Evaluation," *Journal of Soil Science*, 40:477-492, 1989.
17. M.F. Goodchild, G. Sun, and S. Yang, "Development and Test of an Error Model for Categorical Data," *International Journal of Geographical Information Systems*, 6(2):87-104, 1992.
18. P.F. Fisher, "First Experiments in Viewshed Uncertainty: Simulating the Fuzzy Viewshed," *Photogrammetric Engineering and Remote Sensing*, 58:345-352, 1992.
19. P.A. Burrough and A.U. Frank, editors, *Geographic Objects with Indeterminate Boundaries*, Taylor and Francis, London, 1996.

20. P.F. Fisher. "Probable and Fuzzy Models of the Viewshed Operation," in M.F. Worboys, editor. *Innovations in GIS I*, Taylor and Francis, London, pp. 161-175, 1994.
21. P.F. Fisher. "Boolean and Fuzzy Regions," in P.A. Burrough and A.U. Frank, editors, *Geographic Objects with Indeterminate Boundaries*, Taylor and Francis, London, pp. 87-94, 1996.
22. S. Kennedy. "The Small Number Problem and the Accuracy of Spatial Databases." Chapter 16. *The Accuracy of Spatial Databases*, (eds. M. Goodchild and S. Gopal). Taylor and Francis, Basingstoke, UK, 1990.
23. D. Stoms. "Reasoning with Uncertainty in Intelligent Geographic Information Systems." *Proc. GIS 87 - 2nd Annual Int. Conf on Geographic Information Systems*, 693-699. American Soc. for Photogrammetry and Remote Sensing, Falls Church VA, 1987.
24. M.F. Goodchild, D.R. Montello, Peter Fohl, and Jon Gottsegen. "Fuzzy Spatial Queries in Digital Spatial Data Libraries." *Proc. FUZZ-IEEE '98*, Anchorage, AK, pp. 205-210, 1998.
25. C.R. Ehlschlaeger, A.M. Shortridge, and M.F. Goodchild. "Visualizing Spatial Data Uncertainty Using Animation." *Computers and Geosciences*, 23(4):387-395, 1997.
26. V. Robinson and A. Frank. "About Different Kinds of Uncertainty in Geographic Information Systems." *Proc. AUTOCARTO 7 Conference*, 1985.
27. V. Robinson. "Implications of Fuzzy Set Theory for Geographic Databases." *Computers, Environment, and Urban Systems*, Vol. 12, pp. 89-98, 1988.
28. V. Robinson. "Interactive Machine Acquisition of a Fuzzy Spatial Relation." *Computers and Geosciences*, Vol. 6, pp. 857-872, 1990.
29. C. Giardina. "Fuzzy Databases and Fuzzy Relational Associative Processors." Technical Report, Stevens Institute of Technology, Hoboken NJ, 1979.
30. J. Baldwin. "Knowledge Engineering Using a Fuzzy Relational Inference Language." *Proc IFAC Symp. on Fuzzy Information Knowledge Representation and Decision Analysis*, pp. 15-21, 1983.
31. L. Zadeh. "Test-Score Semantics for Natural Languages and Meaning Representation via PRUF." *Empirical Semantics*, (ed. B. Rieger), Brockmeyer, Bochum, GR, pp. 281-349, 1981.
32. V. Cross. "Using the Fuzzy Object Model for Geographical Information Systems." *Proc. 18th Intl. Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, pp. 814-818, June 10-12, 1999.
33. E. L. Usery, "A Conceptual Framework and Fuzzy Set Implementation for Geographic Features." in *Geographic Objects with Indeterminate Boundaries*, ed. P. Burrough and A. Frank, Taylor and Francis, London, pp. 71-86, 1996.
34. Y. Leung. *Spatial Analysis and Planning Under Imprecision*, Amsterdam, Elsevier, 1988.
35. M.H. Katinsky, *Fuzzy Set Modeling in Geographic Information Systems*. Unpublished Master's Thesis, Department of Geography, University of Wisconsin-Madison, Madison, WI, 1994.

36. T. Sarjakoski, "How Many Lakes, Islands, and Rivers are there in Finland: a Case Study of Fuzziness in Extent and Identity of Geographic Objects," *Geographic Objects with Indeterminate Boundaries*, ed. P. Burrough and A. Frank, Taylor and Francis, London, pp. 299-312, 1996.
37. A.U. Frank and M.F. Goodchild, "Two Perspectives on Geographic Data Modelling," Technical Paper 90-11, National Center for Geographic Information and Analysis (NCGIA), 1990.
38. H. Couclelis, "Beyond the Raster-Vector Debate in GIS," in Frank, A.U., Campari, I. And Formentini, U. (Eds), *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Lecture Notes in Computer Science 639, Berlin: Springer, pp.65-77, 1992.
39. V. B. Robinson, "A Perspective on Managing Uncertainty in Geographic Information Systems with Fuzzy Sets," *Proc. FUZZ-IEEE '98*, pp. 211-215, 1998.
40. T.J. Davis. and C. P. Keller, "Modelling Uncertainty in Natural Resource Analysis Using Fuzzy Sets and Monte Carlo Simulation: Slope Stability Prediction," *International Journal of Geographical Information Systems*, 11(5): 409-434, 1997.
41. D. Altman, "Fuzzy Set Theoretic Approaches for Handling Imprecision in Spatial Analysis," *International Journal of Geographical Information Systems*, 8(3): 409-434, 1994.
42. M. Cobb. *An Approach for the Definition, Representation and Querying of Binary Topological and Directional Relationships between Two-Dimensional Objects*, Ph.D. dissertation, Tulane University, New Orleans, LA, 1995.
43. M. Cobb. and F. Petry, "Modeling Spatial Relationships within a Fuzzy Framework," *Journal of the American Society for Information Science*, 49(3). pp.253-266, 1998.
44. M. Cobb. and F. Petry, "Fuzzy Querying of Binary Relationships in Spatial Databases," *Proc. IEEE Intl. Conference on Systems, Man and Cybernetics*, Vancouver, BC, pp. 3624-3629, 1995.
45. M.J. Egenhofer. and R. Franzosa, "Point-Set Topological Spatial Relations," *Intl. J. Geo. Info. Sys.*, Vol. 5, No. 2, pp. 161-174, 1991.
46. M. Cobb. and F. Petry, "Integration of a Fuzzy Query Framework with Existing Spatial Query Languages," *Proc. Fifth IEEE Intl. Conf. On Fuzzy Systems (FUZZ-IEEE)*, New Orleans, LA, pp. 93-99, 1996.
47. M. Nabil, J. Shepherd and A.H.H. Ngu, "2D Projection Interval Relationships: A Symbolic Representation of Spatial Relationships," *Advances in Spatial Databases: 42nd Symposium, SSD '95*, pp. 292-309, 1995.
48. J. Sharma. and D.M. Flewelling, "Inferences from Combined Knowledge about Topology and Direction," *Advances in Spatial Databases: 42nd Symposium, SSD '95*, pp. 279-291, 1995.
49. E. Clementini, J. Sharma and M.J. Egenhofer, "Modelling Topological and Spatial Relations: Strategies for Query Processing," *Computers and Graphics*, Vol. 18, No. 6, pp. 815-22, 1994.

50. H.-P. Kriegel, M. Schiwietz, R. Schneider, and B. Seeger, "Performance Comparison of Point and Spatial Access Methods," in A. Buchmann, O. Gunther, T. Smith & T. Wang (Eds.), *Design and Implementation of Large Spatial Databases*, Lecture Notes in Computer Science 409, Santa Barbara, CA: Springer-Verlag, pp. 89-114, 1989.
51. H. Samet, *The Design and Analysis of Spatial Data Structures*, Reading MA: Addison-Wesley, 1989.
52. S.K. Chang, et al. "An Intelligent Image Database System," *IEEE Transactions on Software Engineering*, Vol. 14, No. 5, pp. 681-688, 1988.
53. J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, Vol. 26, No. 11, pp. 832-843, November 1983.
54. A. Saalfeld, "Conflation: Automated Map Compilation," *International Journal of GIS*, 2(3):217-228, 1988.
55. M. Godau, "A Natural Metric for Curves--Computing for Polygonal Chains and Approximation Algorithms," *Proc. Symposium on Theoretical Aspects of Computer Science*, STACS '91, Springer Lecture Notes in Computer Science, V. 480, pp. 127-136, 1991.
56. D. Siegel, "A non-traditional solution to conflation," *Proceedings of the Urban and Regional Information Systems Association annual conference*, Washington, D.C., 16-20 July, pp. 462-75, 1995.
57. L. A Zadeh, "Fuzzy Sets," *Information and Control*, pp. 338-353, 1965.
58. G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976.
59. M. A. Cobb, M.J. Chung, V. Miller, H. Foley III, F. E. Petry, K. B. Shaw, "A Rule-Based Approach For The Conflation of Attributed Vector Data," *Geoinformatica*, 2(1), pp. 7-35, 1998.
60. H. Foley, F. Petry, M. Cobb and K. Shaw, "Utilization of an Expert System for the Analysis of Semantic Characteristics for Improved Conflation in Geographic Information Systems," *Proc. of the 10th Intl. Conf. On Industrial and Engineering Applications of AI*, Atlanta, GA, pp. 267-275, 1997.
61. H. Foley, F. Petry, M. Cobb and K. Shaw, "Using Semantic Constraints for Improved Conflation in Spatial Databases," in *Proc. 7th Intl. Fuzzy Systems Association World Congress*, Prague, 193-197, 1997.
62. E. Lim, J. Srivastava and S. Shekar, "An Evidential Reasoning Approach to Attribute Value Conflict Resolution in Database Integration," *IEEE Transactions on Knowledge and Data Engineering*, 8(5):707-723, 1996.
63. M. Cobb, F. Petry and K. Shaw, "Uncertainty Issues of Conflation in a Distributed Environment," *Proceedings of GIS/LIS '98*, Fort Worth, TX, pp.436-448, Nov.10-12, 1998.
64. M. Cobb, H. Foley III, R. Wilson, M. Chung and K. Shaw, "An OO Database Migrates to the Web," *IEEE Software*, 15(3), pp. 22-30, May-June.



DEPARTMENT OF THE NAVY
NAVAL RESEARCH LABORATORY
4555 OVERLOOK AVE SW
WASHINGTON D C 20375-5320

IN REPLY REFER TO:
5060
Ser. 1001/015
11 February 1999

Dr. Maria Cobb
16 Quail Hollow Drive
Hattiesburg, MS 39402

Dear Dr. Cobb:

On behalf of the Naval Research Laboratory, I am pleased to inform you that you have been selected, as co-author with Mr. Shaw and Ms. Chung, to receive an Alan Berman Research Publications Award (ARPAD) for 1998. Each year recipients of the award are formally recognized at an awards dinner held in their honor, at which their research contributions are cited; and each federal government recipient is presented a memento of the occasion. We would be pleased if you would be our guest at this year's 31st celebration of the Annual Research Publication Awards and dinner to be held on Friday, 12 March 1999 at the Bolling Officers' Club.

If you plan to attend, the Laboratory will make reservations for you and one guest at no cost to you. Please inform your Division's ARPAD Coordinator or Angie Allison at (202) 767-7225 (allison@utopia.nrl.navy.mil) to ensure that a reservation is confirmed for your party.

Congratulations on your outstanding achievement. We look forward to seeing you on 12 March.

Sincerely,

For the following book chapter
in Succeeding with Object Databases

Explanation of Award - ~~_____~~

Alan Berman Research Publication and Edison Patent Awards

The Annual Research Publication Awards Program was established in 1968 to recognize the authors of the best NRL publications each year. These awards not only honor individuals for superior scientific accomplishments in the field of naval research but also seek to promote continued excellence in research and in its documentation. In 1982, the name of this award was changed to the Alan Berman Research Publication Award, in honor of its founder.

There were 247 separate publications submitted by the divisions in 1996 to be considered for recognition. Of those considered, 38 were selected. These selected publications represent 177 authors, each of whom received a publication awards certificate, a bronze paperweight, and a booklet listing the publications that were chosen for special recognition. In addition, NRL authors share in their respective division's monetary award.

The winning papers and their respective authors are listed below by their research units. Non-Laboratory coauthors are indicated by an asterisk.

NRL also recognizes patents as part of its annual publication awards program. The NRL Edison Patent Award was established in January 1991 to recognize NRL employees for outstanding patents issued to NRL by the U.S. Patent and Trademark Office during the preceding calendar year. The award recognizes significant NRL contributions to science and engineering as demonstrated by the patent process that are perceived to have the greatest potential benefit to the country. Of the 15 patents considered for 1996, 4 were selected, representing 12 inventors. They are listed under the NRL Edison Patent Awards.

Command Support Division

Laser-Heated Radiation Dosimetry Using Transparent Thermoluminescent Glass
Tommy L. Johnson, Brian L. Justus, and Alan L. Huston

Radar Division

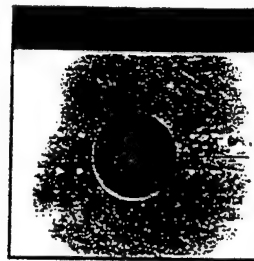
Automatic Recognition of ISAR Ship Images
Scott A. Musman, David W. Kerr, and Charles McKay Bachmann

Digital and Analog Sidelobe Canceller Performance with the AN/SPS-49(V) Radar
Robert M. Crisler, Pete Hansen, John L. Walters, Kevin Luc,
Donald L. Wilson, William L. Thrift, and David A. Alessio

Information Technology Division

Automated Consistency Checking of Requirements Specifications
Constance L. Heitmeyer, Ralph D. Jeffords, and Bruce G. Labaw

*Data-Delay Evaluation in Integrated Wireless Networks Based
on Local Product-Form Solutions for Voice Occupancy*
Jeffrey E. Wieselthier, Craig M. Barnhart, and Anthony Ephremides*



Succeeding with Object Databases

**A Practical Look at Today's
Implementations with Java™
and XML**

Akmal B. Chaudhri
Roberto Zicari

Wiley Computer Publishing



John Wiley & Sons, Inc.

NEW YORK • CHICHESTER • WEINHEIM • BRISBANE • SINGAPORE • TORONTO

The Geospatial Information Distribution System (GIDS)

The widespread use of computers has fostered a progression of mapping from the form of paper charts, maps, and satellite photos to the digital format. The National Imagery and Mapping Agency (NIMA), bearing the sole responsibility for developing mapping information for the Department of Defense, embarked on a program to transform their traditional paper mapping products into digital format. To accomplish this goal, in the 1980s, NIMA designed the Vector Product Format (VPF) as its database specification. VPF was developed as a relational data model. As the VPF products were reviewed and used, however, it became apparent that the relational format was not suited to the complexity of spatial mapping data. For example, the use of many tables to represent spatial topology of one feature resulted in referential integrity problems during an update process.

The Naval Research Laboratory's (NRL) Digital Mapping, Charting, and Geodesy Analysis Program (DMAP) at Stennis Space Center in Mississippi proposed a possible solution to some of the VPF problems. An alternate data model using object-oriented technology seemed to accommodate the complexity of spatial data. In 1994, the DMAP team was able to successfully prototype the first object-oriented application using one of NIMA's VPF products, the Digital Nautical Chart (DNC). The prototype showed the reusability and rapid-prototyping that resulted from use of object-oriented technology. Consequently, DMAP expanded the object-oriented data model to integrate the other VPF products with the DNC into an object-oriented VPF (OVPF). These additional VPF products are Digital Topographic Data (DTOP), Vector Map (VMAP), Urban Vector Map (UVMAP), World Vector Shoreline (WVS), and Foundation Feature Data (FFD). DMAP has continued to advance OVPF to include other NIMA data types such as Raster Product Format (RPF), which is the basic format for raster products such as satellite imagery.



Succeeding with Object Databases

A Practical
Look at Today's
Implementations
with Java™
and XML

Akmal B. Chaudhri
Roberto Zicari



and scanned charts; and Text Product Standard (TPS), which uses SGML-based standard for textual information such as sailing directions.

Having demonstrated that object-oriented technology easily accommodates the complexity of the spatial data, the next milestone for NIMA and DMAP was to provide a vehicle for distribution. As the Web technology began to advance with the rise of the Java programming language and object-oriented standard committees such as the Object Management Group (OMG), DMAP was able to prototype spatial information distribution over the Web. This was through an area-of-interest (AOI) application called the Geospatial Information Distribution System (GIDS). The GIDS provides all available information over the specified AOI independent of the data type. Some of the more diverse data types handled by the GIDS include ESRI's shape file format, video clips, audio clips, and other industry standards such as tiff, gif, and jpeg. Adding these to the GIDS was relatively simple due to the object-oriented nature of the data model.

Much of the work to date has been the database design to accommodate the multiple spatial data types, the development of a Web applet for map display, and the implementation of query mechanisms. While this work has concentrated on two-dimensional mapping, the need for the three-dimensional representation of some geographic entities to support mission preparation and rehearsal especially in an urban setting is also present. Thus DMAP, in conjunction with the University of New Orleans, proceeded to research and design a 3D spatial data model, called VPF+.

This chapter describes the overall system and database design. As an example, an actual experimental situation, the March 1999 Urban Warrior Experiment (UWE), is used to demonstrate the use of GIDS. Furthermore, the VPF+ data model, as well as its applicability in the UWE, is presented.

Use of Object-Oriented Technology and ODBMS

Object-oriented technology was chosen to handle the complexity of spatial data. Some spatial data such as VPF produced by NIMA already is designed as relational data. A minimum of nine tables is used to define one point feature; at minimum, sixteen tables are used to define one polygon feature. Rather than considering a feature as rows of nine to sixteen tables, an object-oriented approach allows data manipulation and management at a feature level. An analogous point feature object would have the same values that are defined from the rows of nine tables, as an example, for a specific feature; however, instead of having to access feature information from different rows of different tables for a given feature, a single access to a feature object would provide all the information for that given feature.

The complexity of spatial data is also due to the stored topology (i.e., immediate neighbor information). When the geometry of a feature changes in any manner, this requires the stored topology to be recomputed. Recomputation implies a change to table columns for a certain row. Due to the dependency on other related tables, recomputation may require several tables to be changed. Recomputation may also affect adjacent feature information. Tables related to the adjacent feature may also be modi-

fied. Such a rippling effect of feature updates makes the manipulation and management of spatial data in relational format error-prone. Maintenance of referential integrity during these updates of multiple features is an additional concern. On the other hand, for feature objects, a change is localized to that feature object only. Thus, a change is applied by accessing the feature object. Likewise, topology recomputation requires those adjacent feature objects to be accessed and modified accordingly.

As the volume of data increased and the need for multi-user access developed, the need to have true database functionalities such as persistent storage, controlled access to the data, and backup and recovery capabilities, among others, became important. Object-oriented database management systems (ODBMS) provide these functionalities specifically for objects.

The distinction between persistent and transient objects is somewhat less clear for ODBMSs than RDBMSs, due to the tightly coupled nature of the object-oriented database with the application. Transient objects are defined as those objects that exist in the computer's memory only during execution of the application, whereas persistent objects exist even when there is no application program running. Transient and persistent objects can coexist within a running process. Persistent data accessed from the database can be assigned to a transient object. It is important for application programs to manage both transient and persistent data in consistent ways so updates to persistent data are made when needed, and data that should remain transient are not inadvertently made persistent.

A natural extension to the decision of using object-oriented technology and an ODBMS was the utilization of the Common Object Request Broker Architecture (CORBA). CORBA allows interoperability among different programming languages as well as different hardware platforms. For example, this project demonstrated interoperability between Smalltalk and Java, and from a Solaris platform to Windows NT using CORBA infrastructure. Since the ODBMS selected provided an Object Request Broker (ORB) directly interfaced to the ODBMS, the data access was achieved at the repository level. Thus, multi-user access was provided through different clients such as applets, which enabled direct Web browser interaction with the ODBMS.

2D GIDS Design

The GIDS has a client/server architecture. It is composed of server, interface, and client modules as shown in Figure 17.1.

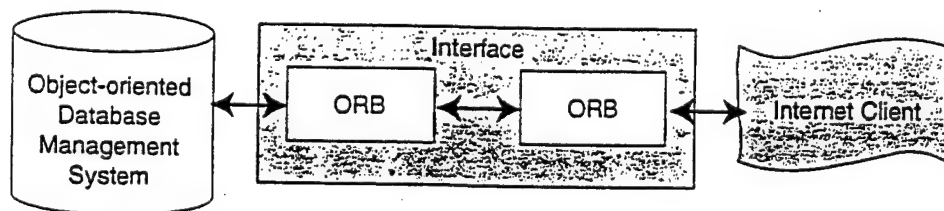


Figure 17.1 GIDS system components.

Server

A commercial, off-the-shelf object-oriented database management system, GemStone, is used as an object server that stores, manipulates, and processes objects referenced by each client. The server consists of two functional modules: data storage and data manipulation or processing. Based on the request from each client, the GemStone server searches and retrieves only those objects that meet the requested criteria. Data search for retrieval is performed mostly on the server.

A server maintains a class hierarchy as shown in Figure 17.2. The MapObject class is the super class or parent to all the spatial classes. Each database has its subclasses. For example, Figure 17.3 shows the class hierarchy for VPFDATABASE and MMDATABASE.

Each subclass of MapObject has a global variable, Databases, which maintains a collection of all its instances. The VPFDATABASE class is the superset of all VPF data, and has a class variable or a global dictionary called Databases that contains all instances of the VPFDATABASE class. A root entry to any feature access begins with the Databases dictionary. Likewise, other classes such as ShapeDatabase has a global variable called Databases. The MMDATABASE class, however, does not have a global variable that maintains a collection of all its instances. This is because the other database classes have a complex data structure such as the complex hierarchical grouping as well as metadata information. This can be seen in Figure 17.4.

For MMDATABASE instances, however, each instance is a multimedia object such as a JPEG object. Thus, rather than maintaining a global variable called Database to hold all the instances, another global variable is used, MMmanager.

The MMmanager is an instance of VPFSpatialDataManager class. The GIDS has implemented a quadtree based on a minimum-bounding rectangle to index all spatial objects. This is based on the regular recursive subdivision of blocks of spatial data into four equal-sized cells, or quadrants. Cells are successively subdivided until some criterion is met, usually either that each cell contains homogeneous data, or that a preset number of decomposition iterations have been performed. Thus, the cells of a quadtree are of a standard size (in powers of two) and are nonoverlapping in terms of actual representation. This can be seen in Figure 17.5.

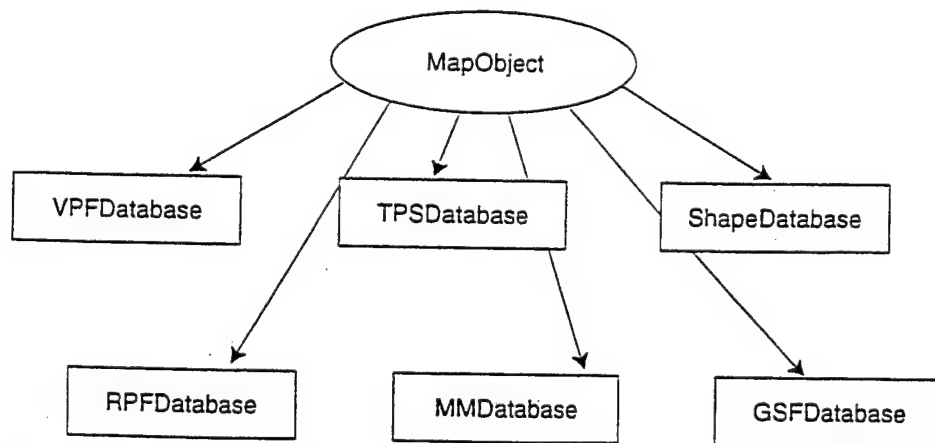


Figure 17.2 GIDS class hierarchy.

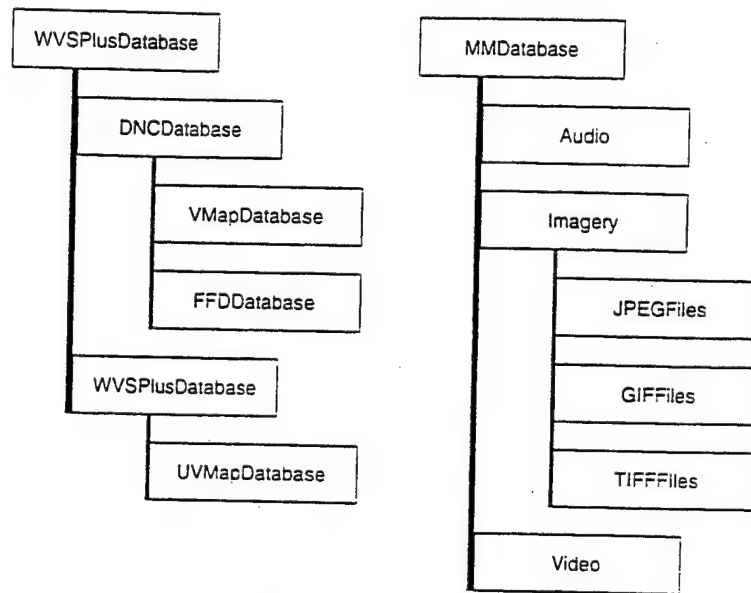


Figure 17.3 GIDS subclass hierarchy.

This spatial indexing scheme is the key to the spatial data integration. All data are spatially indexed into a quadtree. Each database maintains its own quadtree, which allows independent spatial indexing of each common data set. A global spatial integration of all objects in the ODBMS is achieved by addressing all the quadtree instances in the ODBMS. This is how the AOI request is accomplished; the spatial integration enables users to find all information available at the specified AOI. In other words, if, for example, two different data sets from different sources provide information over at San Francisco, California, the GIDS would let the user know that there is information about San Francisco, California from two different data sets that are different in format and contents. This basically achieves the AOI-driven search capability in the GIDS.

Interface

An object request broker (ORB) functions as an interface to the server and client. A server has to have its own broker and a client has to have its own broker. GemORB is a CORBA 2.0-compliant object request broker used by the server. GemORB provides an interface for the Smalltalk programming language. GemORB interfaces directly to the ODBMS. Clients use VisiBroker as their CORBA 2.0 compliant brokers. VisiBroker provides an interface for the Java programming language.

GemORB establishes a connection between the client broker, VisiBroker, to the object server, GemStone, through CORBA compliant communication. See the works by Thomas Mowbray et al., [Mowbray 1997a], the OMG [OMG 1996], and Jon Siegel [Siegel 1996] for information on CORBA. An Interface Definition Language (IDL) file defines a correct mapping of objects between the client and the server. An IDL file also defines operations or methods that are available for clients to invoke on the server.

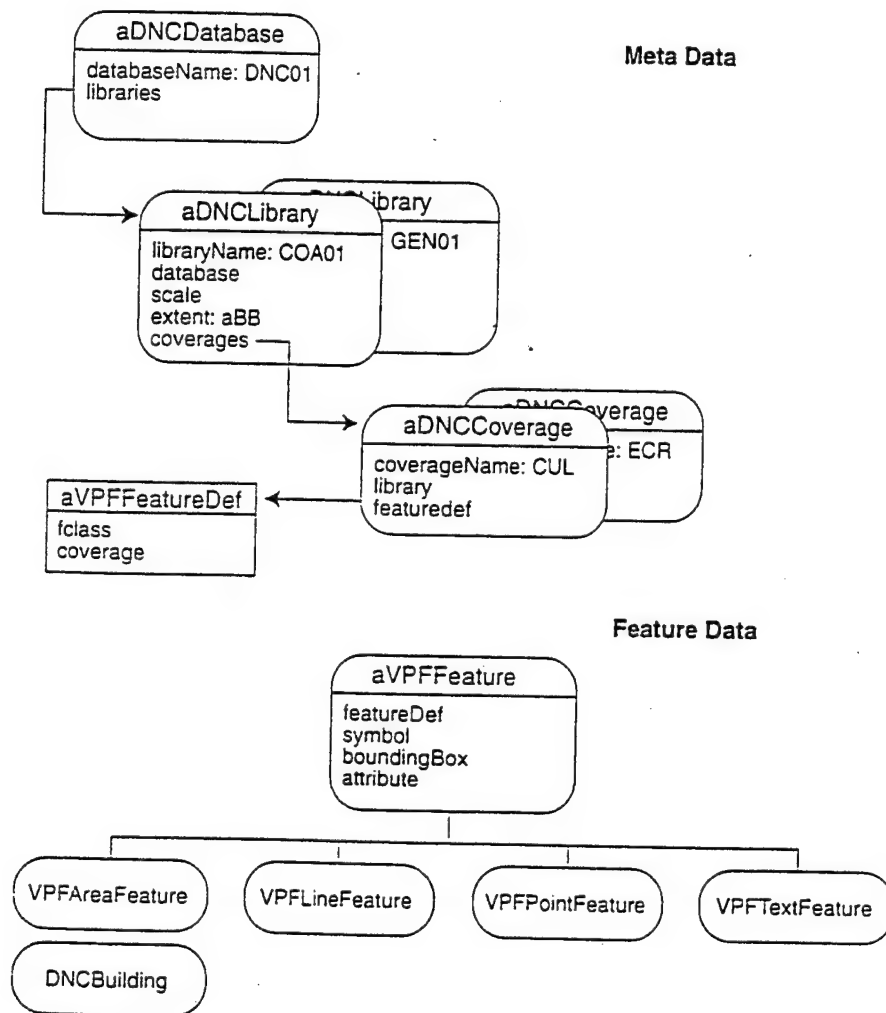


Figure 17.4 VPF data model.

Since GemORB and VisiBroker are based on CORBA, all the benefits of interoperability among programming languages and platforms apply.

An IDL is essential for communication between different programming languages. An IDL file must be created to allow for correct mappings of objects from one application to another; it is the means by which potential clients determine what operations are available and how they should be invoked. In our system, our IDL file defines all of the objects that are common to both client and server, as well as methods that can be invoked to perform certain operations on the server, as shown in Listing 17.1.

The first object that is utilized by both the Java client and the GIDS server is a bounding box for the AOI, an instance of the `BoundingBox` class. To define our `BoundingBox` object, we use a `struct` data type that allows related items to be grouped together. For example, `struct Point {float x,y;};` defines a point to be made up of two float

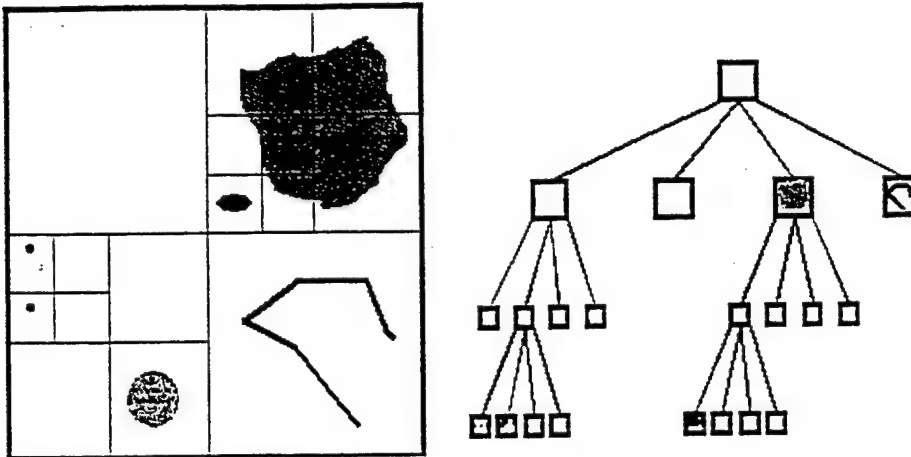


Figure 17.5 Example of a quadtree.

values, x and y . Similarly, our `BoundingBox` is defined to be composed of two points, an origin and a corner: `struct BoundingBox (Point origin; Point corner;)`. We then defined an interface called `GIDBSyms`, which contains the methods on the server that are invoked by the client. An interface is the most important aspect of IDL, since it provides all the information needed for a client to be able to interact with an object on the server.

Note that the interface contains the method names with their parameters, as well as the data type of the returned object. The most complex structure defined in our IDL is the struct `VectorFeature`.

The Smalltalk application has an object class called `VPFFeature` from which our `VectorFeature` is derived. The Smalltalk `VPFFeature` class is more complex. The `VPFFeature` objects have geometry, as well as attributes. The geometry information basically provides the footprint of the object in pairs of latitude and longitude. Attribute information consists of meta-data and actual properties or characteristics of the object. The meta-data provides information about the data, such as VPF specification relevant information as well as information like the originator, security. `VPFFeature` objects are richly attributed; for example, a `DNCBuilding` has attributes such as `bfc`, which is building function code, and `hgt`, for the height of a building. For our Internet applet, though, only those attributes that are needed for display and user queries are defined as shown here.

Our IDL contains another structure that defines `Attribute`: `struct Attribute {string name, value; }`. The `Attribute` structure is used as part of the `VectorFeature` structure and gives attribute names and values for a given feature instance. For example, a given building may have an attribute name "Structure Shape of Roof" with attribute value "Gabled."

The final data type included in our IDL file is a *sequence*, which is similar to a one-dimensional array, but does not have a fixed length. We use sequences to reduce the number of messages passed from server to client. The size of each sequence is determined dynamically on both the server and the client.

```

module GIDBmodule {
    struct Point {float x,y;};
    struct BoundingBox {Point origin,
    struct Attribute {string name,

typedef sequence<Attribute>
typedef sequence<Point> Coordinates;
typedef sequence<string> StringColl;

    struct VectorFeature {
        string      featname;
        long        type;
        AttributeColl attributes
        Coordinates cords
        BoundingBox boundingBox;};

    struct MediaFeature {
        string      objectType;
        string      description;
        string      filename;
        BoundingBox boundingBox;};

typedef sequence<VectorFeature>
typedef sequence<MediaFeature>

    interface GIDBSyms {
        StringColl ReturnDatabasesForAOI (in BoundingBox
        StringColl ReturnCovsAndFeatsForAOI (in BoundingBox aBB, in
            dbname, in string libname);
        FeatureColl ReturnFeats (in StringColl featColl, in aBB
        MediaColl ReturnMediaFeats (in BoundingBox aBB);
    };
}

```

Listing 17.1 IDL used by GIDS.

This IDL file must be compiled on both the client and the server. On the server, the IDL is filed, bindings to objects are made appropriately, and new methods are created. On the Java client, the process is similarly performed via an IDL to Java mapping. Objects defined in the IDL can then be referenced and used in both the client and server code.

Client

A client request for information expects the object server to search and completely process the information. A client therefore receives fully processed information that can be used readily. Fully processed implies that the data are in a useful format by the clients. Once the client receives the requested data, these data are cached on the client

for performance enhancement. Thus, any functionalities that need to be performed using the data, such as spatial query, are all performed on the client.

Clients have an online access to geospatial data such as raster images and vector features over the Internet. These geospatial objects would be retrieved from a GemStone server. Communication between the server and a client is accomplished using CORBA-compliant vendor ORBs. The use of VisiBroker on the client side and GemORB for the database server is completely transparent to anyone accessing the applet. Figure 17.6 shows the basic architecture of our system. A Web-based client has the capability to display, select, and query objects interactively.

The retrieval of features from the GIDS is based on the concept of area of interest (AOI). The first screen of the applet displays a world map from which the user can select a location graphically through the use of a rectangle (bounding box), as shown in Figure 17.7. The user also has the option of entering the coordinates for the AOI manually, or selecting a predetermined region. From the user input, a bounding box of the AOI is transmitted from the applet via CORBA to the Smalltalk server.

The server responds with a set of database and library names for which data are available in that region. NIMA provides VPF data in databases, and each database contains one or more libraries. The user then selects a database and library, resulting in a list of coverages and feature classes being returned from the server through another CORBA request. Finally, the user selects the feature classes of interest and submits a request for them to be displayed, as shown in Figure 17.8. This request results in further CORBA communication, and the server returns a set of all of the features of the requested classes that are located in the given AOI. These features that are returned are complex objects with both geometric (coordinate) and attribute information. The applet can then display, select, and query on the returned features.

The underlying motivation for having a Web-based Java client access our OO mapping database is to give end users the ability to access and use NIMA data quickly and efficiently. At the present time, users of NIMA data must have software to view the data resident on their own computer systems, and must obtain the data primarily on CD-ROM or other storage media with limited Web interaction available for the user. Our Java applet allows any user with a Java-enabled Web browser to access our GIDB

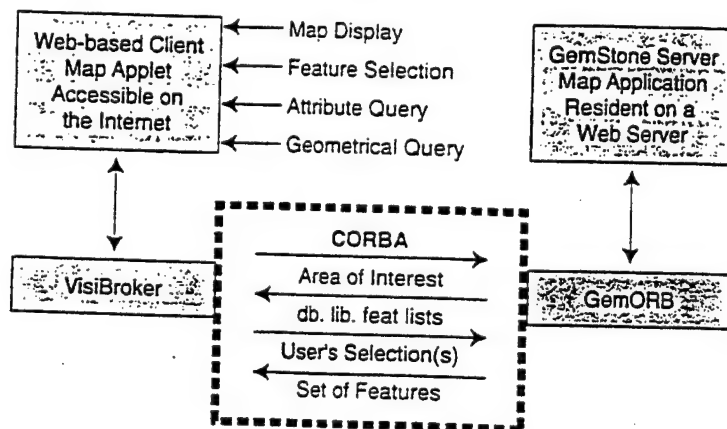


Figure 17.6 Basic system design.

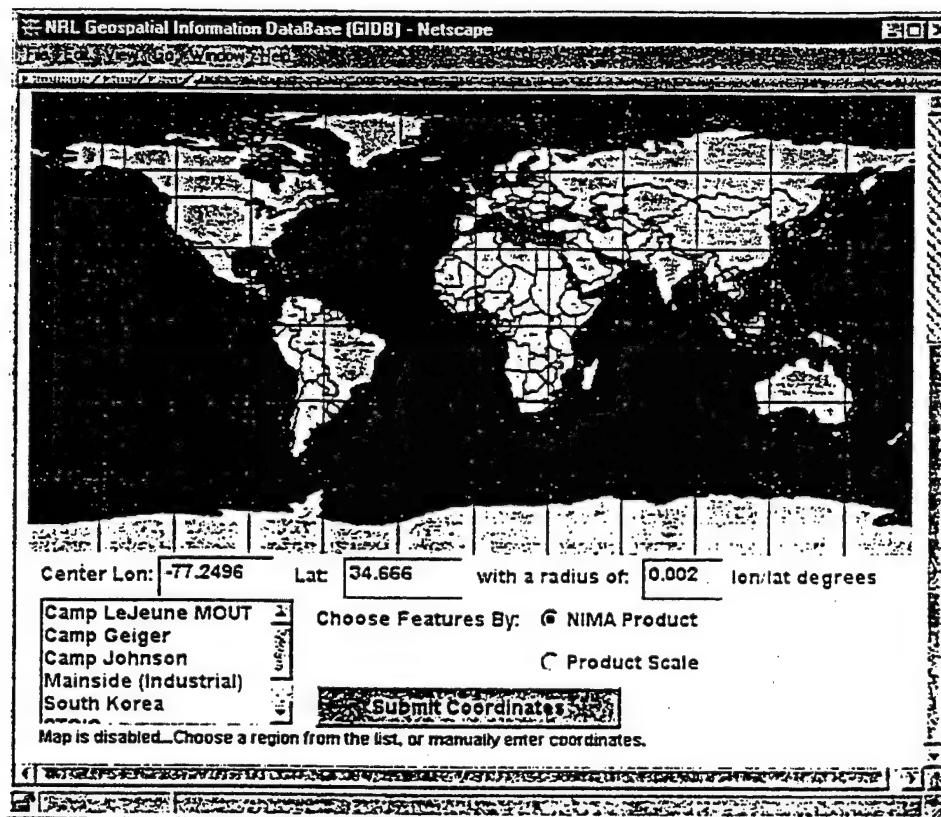


Figure 17.7 GIDS applet map display.

over the Internet and to display map data available in their area of interest. In addition to display of map objects, we have extended the functionality of the Java client to include simple queries, individual feature selection, zoom capabilities, attribute queries, geometrical queries, and updates of attribute values.

After the selected features in a user's AOI have been returned to the Java client and displayed as shown in Figure 17.9, the user can change the colors of the features to distinguish between the feature classes retrieved. A color key is shown providing the color, feature class, and number of those features in the given AOI. The user also has the ability to change the color of the background. Zoom capabilities are provided, allowing the user to zoom in, zoom out, or zoom to a user-specified area in the AOI. An individual feature may be selected by clicking on it in the map pane, resulting in the display of that feature's attributes.

Clicking on the Query button below the map pane performs a simple query. This query lists all of the features in the map pane and gives the user access to each feature's attribute information. More advanced queries can be performed by clicking on the Adv. Query button below the map pane. The advanced query screen allows users to display new feature classes in the AOI. The user can also perform attribute-level queries. For example, the user can highlight all of the four-lane roads, or all buildings

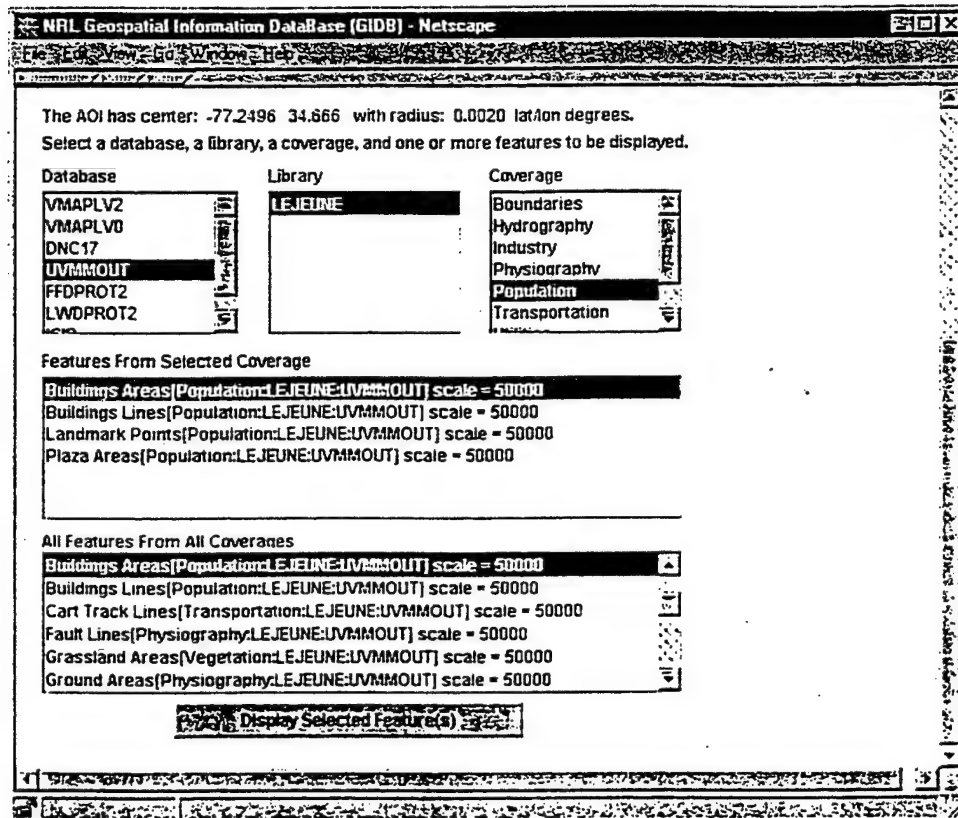


Figure 17.8 User selection window.

that function as government buildings. Users can also perform geometrical queries, such as "find all buildings that are greater than 50 feet from the road," or "find all homes that are within 20 meters of the Embassy."

Update of feature attributes is also possible with the Java client. For example, a newly paved road could have its attribute for surface type updated from "gravel" to "concrete." This function of the applet must be password protected so that only users with authorization can change data in the database.

Another function available in our Web applet includes the ability to perform Internet queries based on our AOI. A user can perform an Internet query by selecting the Internet Query button, and then selecting "Weather," "News," "Yellow Pages," or "Other Maps." For example, if a user decides to find out the weather for the current AOI and selects appropriately, the GIDS server will locate the nearest city to the user's AOI and will open a Web page with that city's local weather forecast.

Our existing Smalltalk mapping application has been extended to the Web utilizing a Java interface via CORBA. The success of our effort is exhibited in the current functionalities of our Java applet on the Web. We have several ongoing projects to improve our Web application, including the display of raster data. We are investigating ways to move to a truly distributed database. Additionally, we want to give users the ability to

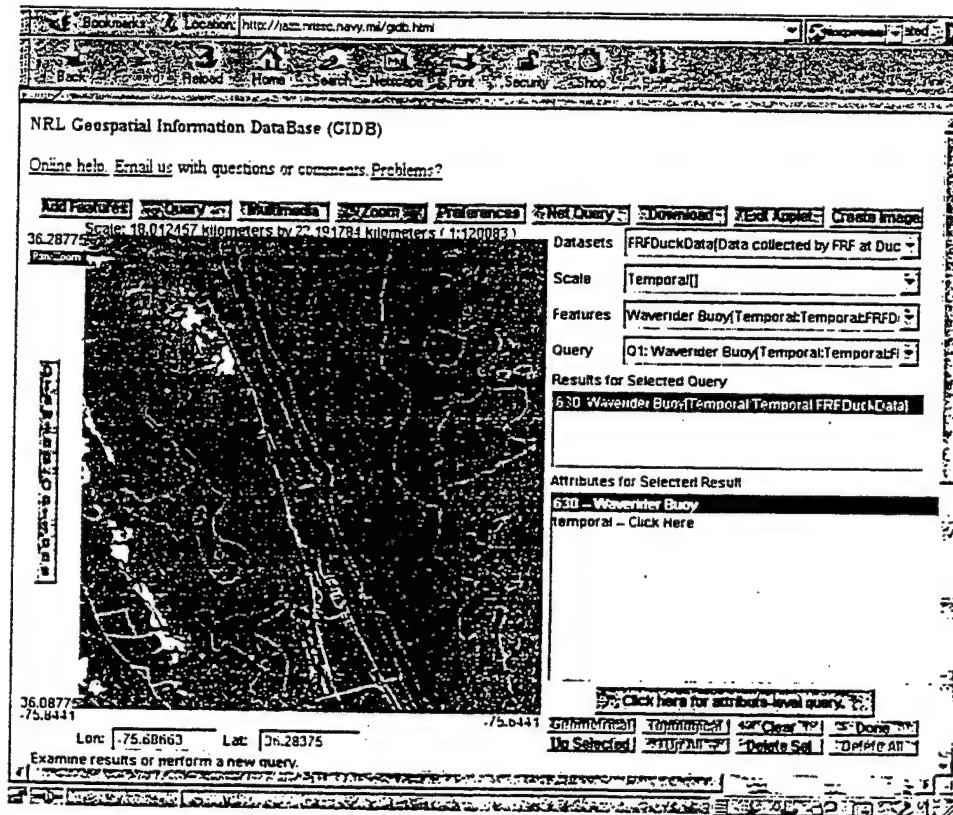


Figure 17.9 Display of selected features.

download data over the Internet from our Java interface to expedite the distribution of mapping data. Another extension to our Java interface is the ability to display the features in our map pane in 3D utilizing VRML. We anticipate the user being able to click on a "Render in 3D" button to obtain a VRML generated 3D model of the features in the current AOI. The open standard of VRML 2.0 is an excellent format for 3D modeling of land and underwater terrain, natural features, and man-made features. We will generate 3D models using gridded, TIN (Triangulated Irregular Network), and vector data. Our VRML models will provide additional information about the AOI by immersing the viewer into and allowing interaction with a virtual world.

Once these tasks are accomplished, users interested in a wide variety of mapping data will be able to access and benefit from our GIDS over the Internet from any platform using a Java-enabled Web browser. This will allow the functionality of more powerful server machines to be exhibited on less capable client machines. It will also give users faster access to mapping data. Our migration to a Web-based mapping client is a revolutionary way of allowing clients with modest computing resources user-friendly access to state-of-the-art mapping data and software.

2D GIDS Application

In this section we discuss the 2D GIDS Application. We begin with a discussion of Urban Warrior and present the IMMAGCS Architecture in Figure 17.10.

Urban Warrior

In the post-cold-war era there has been more focus on urban warfare. The presence of civilians, urban infrastructures, and open space with obstructed views are some of the complications present in urban warfare. In preparation for potential urban warfare, the Marine Corps Warfighting Lab (MCWL) has performed an experiment and demonstration on how to conduct combat in urban settings using the latest technology. In 1997, an exercise called Hunter Warrior took place with the focus on fighting smarter, using less physical force, and relying more on microchips [CNN 1997].

Since many military operations take place via a chain of command, MCWL focused on the command and control activity within the Enhanced Combat Operations Center (ECOC). A specific requirement was imposed in supporting the experiment; the overall system was required to be object-oriented. MCWL believed that an object-oriented system would better meet the overall objective in effectively controlling the urban warfare. The Integrated Marine Corps Multi-Agent Command and Control System (IMMACCS) consists of a number of different components. A real-time display was required to visualize the common tactical picture by officers in the ECOC as activities took place in the "battle space." Stanford Research Institute (SRI) developed and pro-

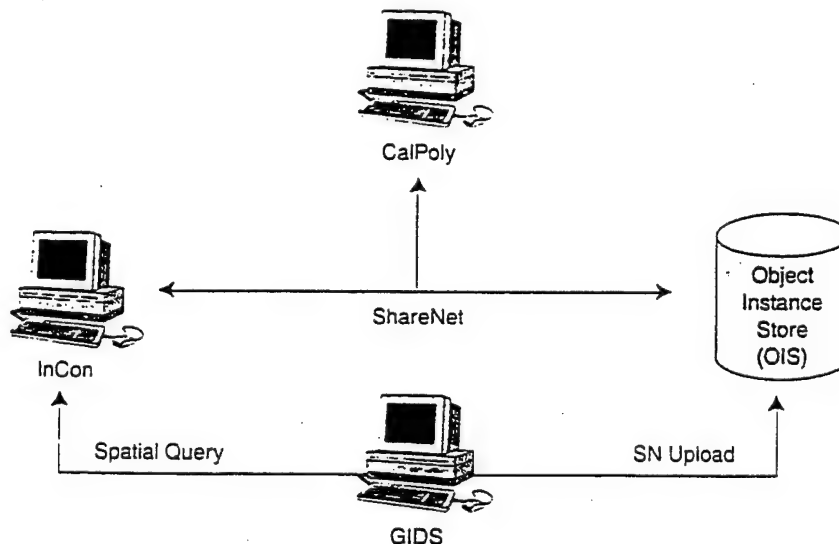


Figure 17.10 IMMAGCS architecture.

vided the two-dimensional (2D) view of the urban battle space. Jet Propulsion Laboratory (JPL) provided a backbone (ShareNet) for all communication among the IMMACCS participants. Common Object Request Broker Architecture (CORBA) was the underlying medium that was used to exchange information among different components of the IMMACCS. California Polytechnic Institute (CalPoly) developed and provided the intelligent "software" agents to assist in the decision making process at the ECOC. SPAWAR's Multi-C4I Systems IMMACCS Translator (MCSIT) ingested all track information from legacy command and control systems such as Joint Maritime Command Information System (JMCIS) and translated it into object-oriented format for the IMMACCS components to access and use. SRI's 2D viewer (InCon) as well as CalPoly's agents required the urban infrastructure to provide a visualization of the battle space as well as a capability to reason about the surroundings.

Both the intelligent agents and the display had two categories of information: static and dynamic. Static information is geospatial information that encompasses physical and built-up environments, man-made structures (e.g., buildings, facilities, and infrastructure), and natural structures (e.g., topography, vegetation, coastlines). Dynamic information deals with tracking the movements of troops, tanks, helicopters and a company of marines, and is based upon the urban infrastructure in terms of its position, mobility, and operation. It is the static information contained in maps that provides much of the strategic and tactical basis for any military operation. Since NIMA's mapping products provide mapping data in relational form and MCWL specifically required the overall system to be object-oriented, DMAP provided the conversion to object-oriented format through the GIDS. The GIDS was used as the geospatial component of the IMMACCS system.

Dynamic as well as static urban infrastructure objects were persisted in the Object Instance Store (OIS) maintained by ShareNet. All objects must be in the OIS to be accessible by each system component. The OIS stores only the attributes of urban infrastructure objects, not the positional information. Since the InCon 2D viewer did not support vector maps for the infrastructure objects, an image was used as a reference map. Therefore, InCon needed to query the GIDS to determine which objects were available in the area of interest (AOI). Both the GIDS and the OIS maintained a global identification of each infrastructure object. When the GIDS provided the global identification of the objects to InCon, InCon then in turn requested OIS for other information. This two-step query process was implemented because the attributes of the infrastructure objects as provided by NIMA are a subset of the attributes defined for each infrastructure object in the IMMACCS object model (IOM). The OIS provides more information relevant to the IMMACCS environment. Due to the imposed requirement of using object-oriented systems, CORBA readily was realized among different systems to create an integrated system.

The following list of GIDS capabilities were provided to the IMMACCS as a part of the integrated system:

- Transform the relational vector map information to object-oriented format
- Upload the urban infrastructure objects to the ShareNet's Object Instance Store via CORBA
- Allow InCon to perform spatial query via CORBA

Figure 17.10 shows the overall IMMACCS and the GIDS support within the IMMACCS.

Additional functionality was tested during the Urban Warrior. An online spatial data updating took place from Bethesda, Maryland, to San Diego, California. This was a valuable test, which demonstrated that the object-oriented technology allows ease of updating complex data, such as spatial data. This work is discussed in detail in [Chung 1999].

3D GIDS

Mapping has been the chief means of geospatial data visualization provided by traditional Geospatial Information Systems (GIS). A GIS can produce a highly accurate digital map for a given area of interest, using well-recognized symbols to represent such features as mountains, forests, buildings and transportation networks. Although this flat view of the world provides an excellent means of orienting the user to the general nature and location of the features for a given area, it fails to provide the full experience that comes from viewing a three-dimensional (3D) environment. To address this shortcoming, NRL's DMAP, in conjunction with the University of New Orleans' Computer Science department, has investigated the development of a 3D-GIS that would assist the U.S. Marine Corps with mission preparation and rehearsal and also provide on-site awareness during actual field operations in urban areas.

We designed our 3D-GIS to supplement NIMA's traditional 2D digital-mapping output with a 3D synthetic environment counterpart. The map output remains useful for general orientation and query functions. The 3D output addresses the targeted application area. Instead of merely applying photo-textures to highly simplified geometric shapes, we include detailed, 3D, natural and man-made features such as buildings, roads, streetlights, and so on. We maximize the user's experience in this synthetic environment by providing for movement within and interaction with the environment consistent with the types of interactions expected of marines during anticipated urban operations. We construct the environment such that the user can *walk* or *fly* across terrain and can *walk* into buildings through doorways or *climb* through open windows. Since we construct synthetic buildings that conform to their real world floor plans, direct *line of sight* into and out of buildings through open doorways and windows is available. Additionally, once inside a building, the user can walk through interior rooms via interior doorways and climb stairs to different floors.

Our 3D synthetic environment is constructed using an extension of the NIMA's Vector Product Format (VPF) [DoD 1996] designed by DMAP and the University of New Orleans. The extended VPF, referred to as VPF+ [Abdelguerfi 1998], makes use of a nonmanifold data structure for modeling 3D synthetic environments. The data structure uses a boundary representation (B-rep) method. B-rep models 3D objects by describing them in terms of their bounding entities and by topologically orienting them in a manner that enables the distinction between the object's interior and exterior. Consistent with B-rep, the representational scheme of the proposed data structure includes both topologic and geometric information. The topologic information encompasses the adjacencies involved in 3D manifold and nonmanifold objects, and is described using a new, extended Winged-Edge data structure. This data structure is referred to as "Non-Manifold 3D Winged-Edge Topology."

VPF+

The data structure relationships of the Non-Manifold 3D Winged-Edge Topology are summarized in the object model shown in Figure 17.11. References to geometry are omitted for clarity.

There are five main VPF+ primitives:

Entity node. Used to represent isolated features.

Connected node. Used as endpoints to define edges.

Edge. An arc used to represent linear features or borders of faces.

Face. A two-dimensional primitive used to represent a facet of a three-dimensional object such as the wall of a building, or to represent a two-dimensional area feature such as a lake.

Eface. Describes a use of a face by an edge.

Inside the primitive directory, a mandatory Minimum Bounding Box (MBB) table (not shown in Figure 17.11) is associated with each edge and face primitive. Because of its simple shape, an MBB is easier to handle than its corresponding primitive. The primitives shown in Figure 17.11, except for the eface, have an optional spatial index. The spatial index is based on an adaptive-grid-based 3D binary tree, which reduces searching for a primitive down to binary search. Due to its variable length records, the connected node table has a mandatory associated variable length index.

The ring table identifies the ring forming the outer boundary and all internal rings of a face primitive. This table allows (along with the face table) the extraction of all of the edges that form the outer boundary and that form the internal rings of a face prim-

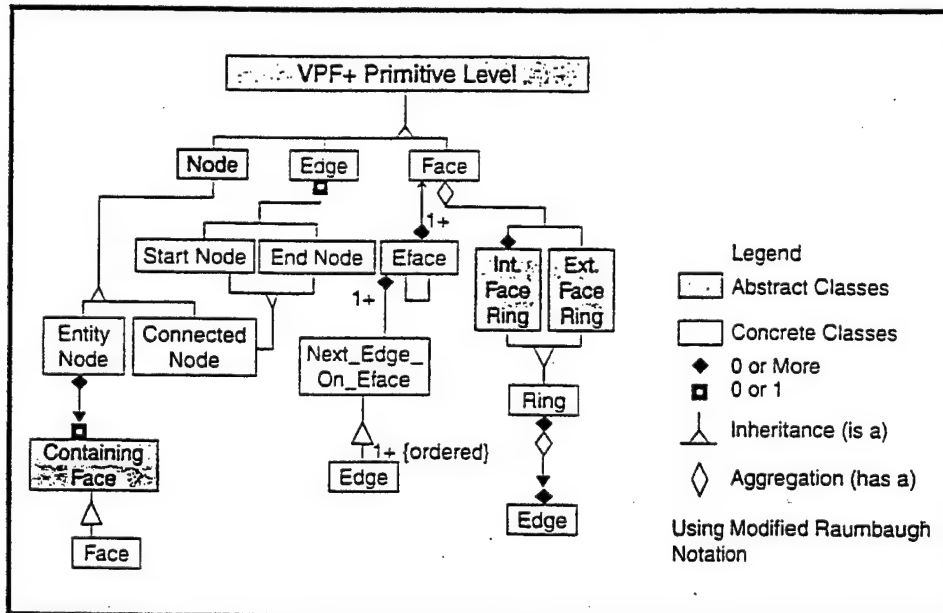


Figure 17.11 VPF+ primitive level object model.

itive. The entity node and the internal and external rings are not essential to the understanding of the VPF+ data structure, and as such, will not be discussed further.

The eface is a new structure that is introduced in VPF+ to resolve some of the ambiguities resulting from the absence of a fixed number of faces adjacent to an edge. Efaces describe the use of a face by an edge and allow maintenance of the adjacency relationships between an edge and zero, one, two, or more faces incident to an edge. This is accomplished by linking each edge to all faces connected along the edge through a circular linked list of efaces. As shown in Figure 17.12, each eface in the list identifies the face with which it is associated, the next eface in the list, and the "next" edge about the face with which the eface is associated. Efaces are also radially ordered in the linked list in a clockwise direction about the edge. The purpose for the ordering is to make traversal from one face to the radially closest adjacent face a simple list operation.

Additionally, VPF's Connected Node Table is modified to allow for nonmanifold nodes. This requires that a node point to one edge in each object connected solely through the node and to each dangling edge. This allows the retrieval of all edges and all faces in each object, and the retrieval of all dangling edges connected to the nonmanifold node.

Unlike traditional VPF's Level 3 topology, the "universe face" is absent in VPF+'s full 3D topology since VPF+ is intended primarily for 3D modeling. Additionally, since 3D modeling is intended, faces may be one- or two-sided. A two-sided face, for example, might be used to represent the wall of a building with one side used for the outside of the building and the other side for the inside of the building. Feature attribute information would be used to render the two different surface textures and color. A one-sided face might be used to represent a portion of a terrain surface. Orientation of the interior and exterior of 3D objects is organized in relation to the normal vector of faces forming the surface boundary of closed objects. Faces may also be embedded within a 3D object.

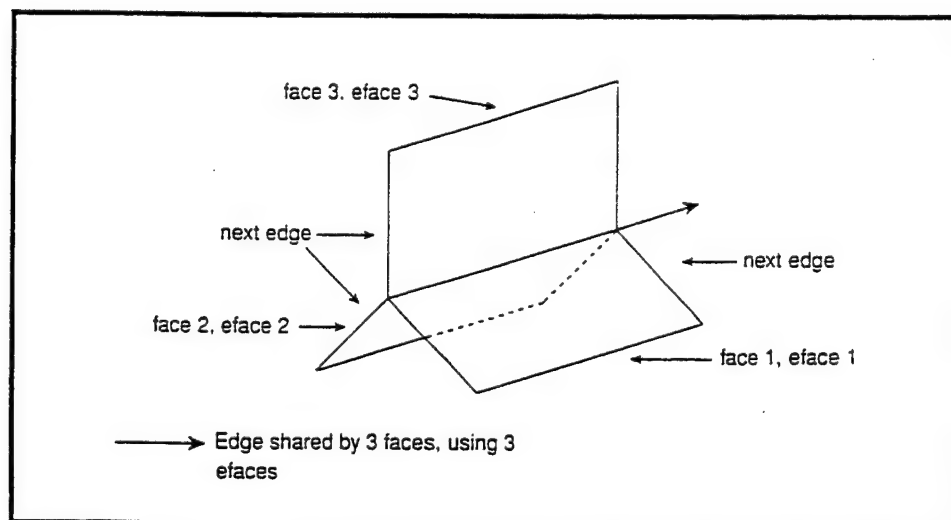


Figure 17.12 Relationship of a shared edge to its faces, efaces, and next edge about each face.

3D GIDS System Overview

Figure 17.13 shows the flow chart of the steps taken to develop the VPF+ database for the United States Public Service Health Hospital in Presidio, California for the Urban Warrior project. Flat floor plans of the building were the only required inputs to this process. These plans provided detailed information about the building such as floor layouts, dimensions, and means of entry. One of the VPF+ tools we have developed is an on-screen digitizer that allows a user to interface with scanned floor plans to extract 3D geometric data and automate production. This allows accurate definition of the overall exterior of the building and also accurate placement of interior rooms, windows, and doorways by defining the nodes, edges, and faces that form the three-dimensional structure of the building. Using this tool and scanned floor plans of the hospital, 3D data were gathered for the building and populated the VPF+ database.

An example of the result of this process is shown in Figure 17.14. Figure 17.14(a) illustrates an example of a typical 2D representation of the building as might be found in a traditional VPF database. Figures 17.14(b) and 17.14(c), on the other hand, show detailed VPF+ object models.

An example of the user interface is shown in Figure 17.15. User interaction is through a Web browser equipped with a 3D graphic plug-in such as Cosmo Player, and an application applet. Interactions with the 3D virtual world include the ability to walk into the building through doorways or climb through open windows, to look through open doorways and windows either from outside or inside the building, and to enter rooms through interior doorways and climb stairs to different floors. A 2D map of the hospital is displayed adjacent to the 3D counterpart in order to provide general orientation.

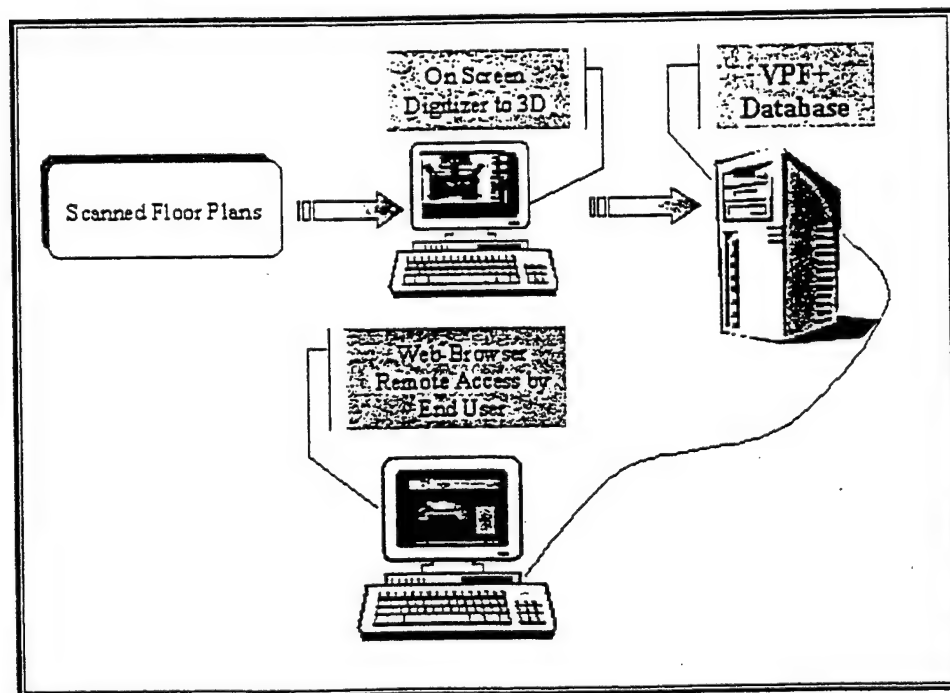


Figure 17.13 Flowchart of 3D model generation for Urban Warrior.

The USPS Hospital is a complex building consisting of nine floors and hundreds of rooms. To provide general orientation to marines inside the building, the 3D model is supplemented with a 2D map of the building. A pointer on the 2D map shows the user's location within the building and the direction in which he or she is heading. This can be seen in Figure 17.16. As the user moves between floors, the 2D map is updated automatically to show the level the user is on.

Experience Report

Through the experience of using an ODBMS and object-oriented technology, we have learned that memory management, input/output (IO) processing, data structure requirements, and utilization of fundamental object-oriented design all play a significant role in the overall performance of the ODBMS. During the development phase of our system, we encountered multiple instances that highlight the importance of giving attention to each of these factors. In this section, we will provide several of our experiences with these factors and will describe what we did to improve our system performance.

Memory management for data storage is a mandatory task because the storage requirement for objects can be significantly greater compared to the relational format. Objects require more storage for NIMA formats because of the requirement to export the updated data back into the original source format. To accomplish this, some of the source information or the relational formats are captured in the object definition to ensure the capability to export. This adds to the increase in the storage requirement. We believe that the storage increase would reduce if the relational information were

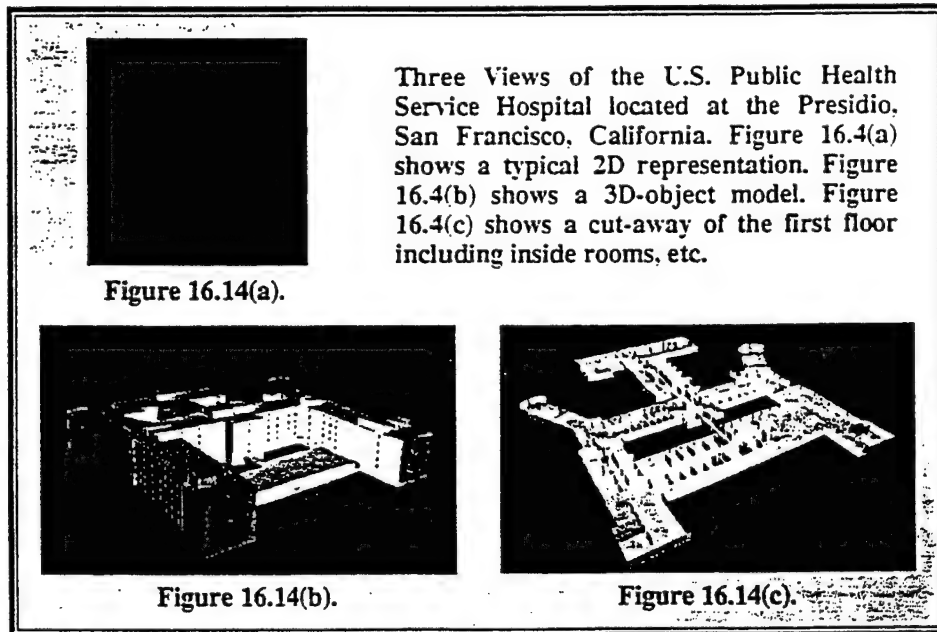


Figure 17.14 The U.S. Public Health Service Hospital at the Presidio.

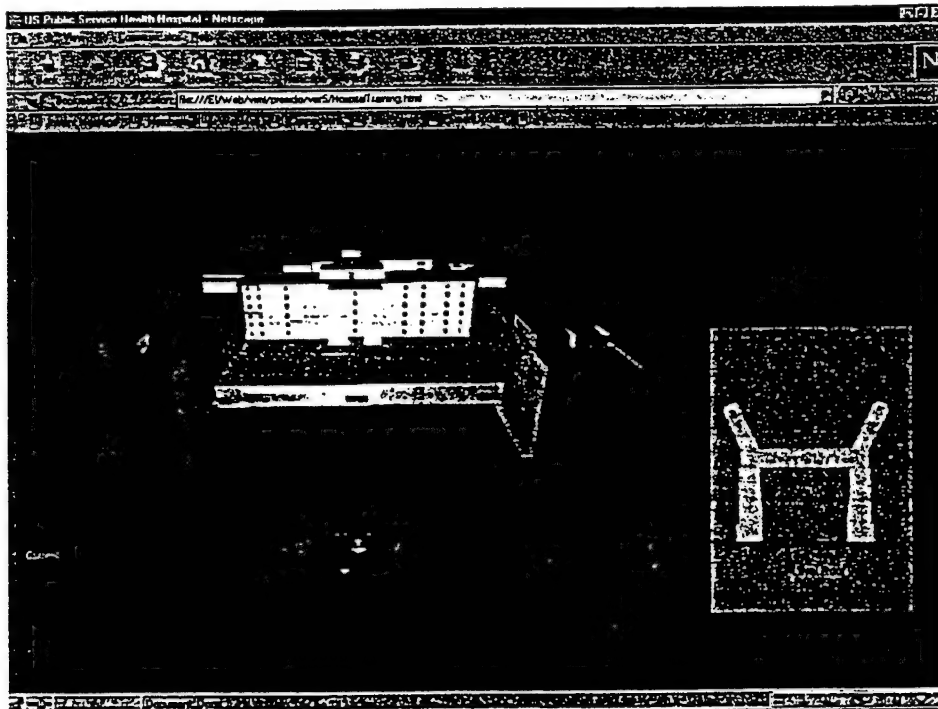


Figure 17.15 Sample user interface to the USPS Hospital.

dropped from the object definition, but our requirement for export capability prevents us from doing so. Consequently, we focus on other ways to minimize storage requirements. The most dramatic improvement in the size of the database occurred when we did not predefine the size of collections. Previously we would use "OrderedCollection new: 200" with the thought that it would be faster to populate the collection if its size were preset. However, the ODBMS would utilize the space for the collection even if only a few objects were actually placed in it. When we changed to using the command "OrderedCollection new" instead, the database size decreased dramatically (over ten-fold) with no major degradation of performance.

Memory management in terms of RAM is also extremely important for increased performance. While testing the initialization of new datasets in our database, we noticed that performance tended to degrade with time; the first few objects initialized quickly, but as more objects were initialized the longer it took for individual object initialization. We concluded that this was due to an increasing number of objects to be managed by RAM before being persisted in the database repository. To resolve this performance degradation, we decided to commit or persist the objects in the database repository during the initialization periodically rather than all at once at the end of initialization. We chose to persist 100 objects at a time, which eliminated this performance problem.

Because network-based client-server applications have inherently poorer performance than single-system applications, we sought to fine-tune the application performance to make the network hit less noticeable. Since IO processing is the most expensive processor

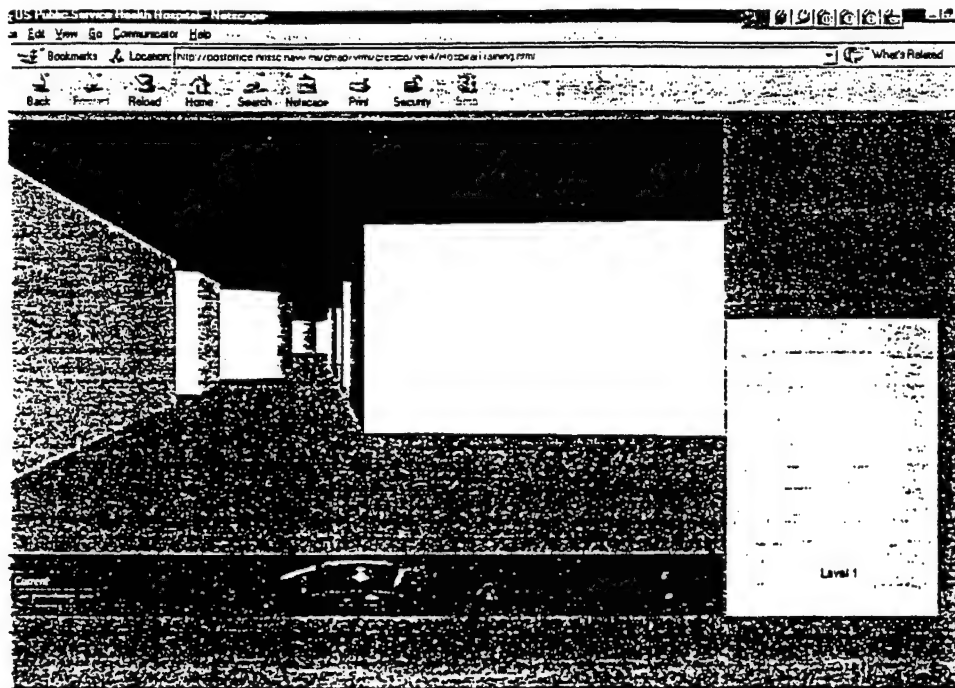


Figure 17.16 Tracking a marine inside the USPS Hospital.

operation, performance in transforming the relational into object format took a significant amount of time. This performance drawback was due primarily to the file open and close operations. Performance improved tremendously when we revised the procedure to read all the contents of a file at one time, storing the data in memory as a dictionary or collection for subsequent use. This memory is released once the data have been used for object creation.

In addition to IO performance tuning, we realized performance improvements after targeting three other areas for revision:

- Reimplementing duplicate, lower-level methods as a single method in a common superclass
- Storing intermediate results in local variables, rather than having repeated message sends
- Reducing the reliance on collections and dictionaries

Because the application was incrementally expanded to handle additional data types, reuse of existing code was not always optimal. This was due largely to the fact that the size of the application made it difficult for developers to "see the forest for the trees." When the time was found to take a broad-based view of the design, it was found that many methods were significantly overlapping in functionality. Merging these methods from the class hierarchy into one method at the superclass level resulted in significant performance improvement. With regard to the second area of performance improve-

ment, we found that often within the same method, message sends were being repeated to compute the same results multiple times. Performance improved when local variables were used to store intermediate results rather than repeating the message sends.

Finally, the use of collections and dictionaries was dramatically decreased. A great performance degradation was noticed when a dictionary size increased from 1000 to 1001 and thereafter. This is due to the Smalltalk hashing functionality. When an item has to be added to a full dictionary, the dictionary is split into two. A new dictionary is created that is 150 percent of the size of the original dictionary. The contents of the dictionary are then copied to a new dictionary. The performance of this process began to degrade significantly as the size of a dictionary reached over 1000. Thus, we have implemented a large dictionary that basically maintains a collection of dictionaries of size 1000. A new dictionary of size 1000 is created each time a new dictionary is needed.

In our ODBMS, each object is assigned a unique object identifier known as an object-oriented pointer (oop). The oop can be used to access any object in the database quickly. We make extensive use of the oop for our Web interface. Our Web interface is a Java applet embedded in an html document and accessible over the Web. The applet communicates with the ODBMS using CORBA. Users of our Web interface want to obtain our data over the Web quickly. To accomplish this, objects in the ODBMS are accessed by the applet through utilization of the object oop. In this manner, we take advantage of the oop to obtain the information that we need from the database, thus quickly providing data to the user.

Conclusion

In this chapter, we have shown how a Web-based distributed system for retrieval and updating of mapping objects has implemented the GIDS. The GIDS architecture relies heavily upon object technology and includes a Smalltalk server application interfaced to a GemStone ODBMS, Java/applet-based client applications, and CORBA middleware in the forms of VisiBroker and GemORB. The GIDS was the realization of our goal to have NIMA data available for electronic information distribution and updating, and played a significant role in the Marine Corps' Warfighting Lab's Urban Warrior Advanced Warfighting Experiment. The architectural components of the system worked well together; using Smalltalk as the server development environment allowed us to prototype new capabilities quickly, while Java provided the Web-based capabilities for the user interface. CORBA proved an excellent choice to serve as a bridge between the two.

A description and prototype of a 3D synthetic environment using VPF+ was also discussed. The 3D developments demonstrated how marines could utilize this technology for an improved situational awareness and mission planning. Users have the ability to view the environment in a more realistic manner. VPF+ is the vehicle that allowed the synthetic environment to be constructed with topology intact. Furthermore, such 3D visualization is Web-enabled through the Web browser plugins. Future directions consist of bridging the gap between the 2D and 3D by allowing 3D rendering from the GIDS's 2D display.

Publisher: Robert Ipsen
Editor: Theresa Hudson
Developmental Editor: Kathryn A. Malm
Managing Editor: Angela Smith
Text Design & Composition: Benchmark Productions, Inc.

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in initial capital or ALL CAPITAL LETTERS. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This book is printed on acid-free paper. ☺
Copyright © 2001 by Akmal B. Chaudhri and Roberto Zicari. All rights reserved.
Published by John Wiley & Sons, Inc.
Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ@WILEY.COM.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

Library of Congress Cataloging-in-Publication Data:

Chaudhri, Akmal B.

Succeeding with Object databases : a practical look at today's implementations with Java and XML / Akmal B. Chaudhri, Roberto Zicari

p. cm.

Includes bibliographical references and index.

ISBN 0-471-38384-8 (cloth : alk. paper)

1. Object-oriented databases. 2. Java (Computer program language) 3. XML (Document markup language) I. Zicari, Roberto. II. Title.

QA76.9.D3 C3848 2000
005.757-dc21

00-043455

Printed in the United States of America.
10 9 8 7 6 5 4 3 2 1

Laboratory there. Dr. Wileden received a degree in mathematics and the MS and PhD degrees in computer and communications sciences from the University of Michigan, Ann Arbor. His current research interests center on tools and techniques supporting seamless integration of advanced capabilities into computing systems. Professor Wileden can be reached at wileden@cs.umass.edu.

Chapter 15: Experiences Using the ODMG Standard in Bioinformatics Applications

Norman W. Paton is a Professor of Computer Science at the University of Manchester, where he co-leads the Information Management Group. He obtained a BSc in Computing Science from Aberdeen University in 1986, and a PhD from the same institution in 1989. From 1989 to 1996, he worked as a lecturer at Heriot-Watt University. His research interests have mainly related to object databases, including work on deductive, active and spatial facilities, and user interfaces to data intensive systems. He is currently also working on bioinformatics, including systems for supporting querying over distributed bioinformatics resources. He can be reached at norm@cs.man.ac.uk.

Chapter 16: An Object-Oriented Database for Managing Genetic Sequences

Zohra Bellahsene is an Assistant Professor in Computer Science at the University of Montpellier II France, since 1987. She received her PhD degree in Computer Science from the University of Paris VI, in 1982, and her Habilitation à Diriger des Recherches from the University of Montpellier II, in 2000. She has devoted her recent research to object-oriented database views and view adaptation in data warehousing systems. Dr. Bellahsene can be reached at bella@lirmm.fr.

Hugues Ripoche has a PhD from Montpellier University / LIRMM laboratory. His interests include object-oriented databases, genetic sequence analysis, and data mining. He currently works at Fi SYSTEM, a group specialized in providing global internet/intranet solutions for dot coms and other companies. Dr. Ripoche can be reached at hugues.ripoche@fisystem.fr.

Chapter 17: The Geospatial Information Distribution System (GIDS)

Miyi Chung has been technically leading the development of the Geospatial Information Database System at Naval Research Laboratory. Her research interests are in spatial data management, spatial data mining in object-oriented database, and spatio-temporal data modeling. She received her BSEE from University of Maryland in College Park in 1985, MSEE from George Washington University in 1991, and pursuing PhD at Tulane University in Computer Science. She can be reached at chung@nrlssc.navy.mil.

Ruth Wilson is a mathematician for the Naval Research Laboratory. Her research interests include object-oriented programming, mathematical techniques to improve digital

mapping, and distributed communication of mapping data between servers and users in real time. Wilson received a BS in mathematics from the University of Southern Mississippi and an MS in mathematics from McNeese State University. She can be reached at ruth.wilson@nrlssc.navy.mil.

Roy Ladner is a computer scientist at the Naval Research Laboratory at Stennis Space Center, Mississippi where he is engaged in research in the generation of 3D synthetic environments. He holds a Master's Degree in Computer Science, and he is currently a PhD candidate in Engineering and Applied Sciences at the University of New Orleans. He can be reached at rladner@nrlssc.navy.mil.

Todd Lovitt is a computer scientist and mathematician with Planning Systems Incorporated. He has been working with the Naval Research Laboratory on the design and development of object-oriented databases of digital mapping data. His research interests include realistic 3-D visualization of urban areas, and distributed techniques for integration and display of disparate geospatial data types across the internet. He received a BS in mathematics and computer science from Mississippi State University. He can be reached at todd.lovitt@psislidell.com.

Maria A. Cobb is an assistant professor in the Department of Computer Science & Statistics at the University of Southern Mississippi in Hattiesburg, MS. Dr. Cobb received a PhD in Computer Science from Tulane University in 1995. She is currently an assistant professor of computer science at the University of Southern Mississippi, and was previously employed by the Naval Research Laboratory as a computer scientist. Her primary research interests are spatial data modeling, including techniques for modeling and reasoning about spatial data under uncertainty, and distributed object-oriented systems. Dr. Cobb can be reached at maria.cobb@usm.edu.

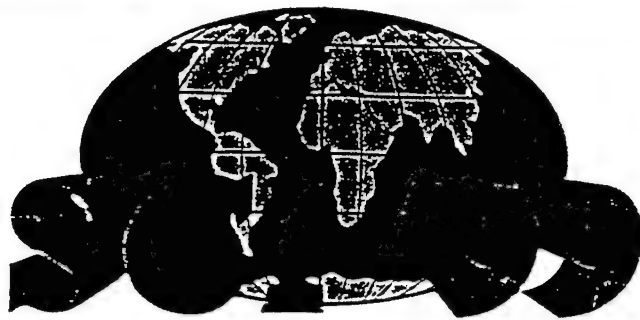
Mahdi Abdelguerfi is chair of the Computer Science Department at the University of New Orleans. His research interests include databases, 3D synthetic environments, and GIS systems. He can be reached at mahdi@cs.uno.edu.

Kevin B. Shaw leads a Naval Research Laboratory R&D team that focuses on advanced geospatial modeling and database design and implementation for improved Naval mapping. Mr. Shaw received a BS in Electrical Engineering from Mississippi State University in 1984, an MS in Computer Science from the University of Southern Mississippi in 1987, and a MEE in Electrical Engineering from Mississippi State University in 1988. Mr. Shaw can be reached at shaw@nrlssc.navy.mil.

Chapter 18: Architecture of the Distributed, Multitier Railway Application DaRT

Juergen Zimmermann is a recognized expert in object technology and in particular in database systems and middleware. As a researcher he was involved in many research projects. For instance he co-operated with the labs of Texas Instruments, Dallas, in the Open OODB project sponsored by DARPA. Between 1996 and 1998 Zimmermann set up the consulting group for Object Design, Germany. In 1998 he joined sd&m as a senior consultant for large-scale projects. His main focus is on software architectures and IT strategies.

World Multiconference on SYSTEMICS, CYBERNETICS AND INFORMATICS



and
ISAS '99

(5th International Conference on Information Systems
Analysis and Synthesis)

July 31- August 4, 1999
Orlando, Florida

PROCEEDINGS

Volume 1

INFORMATION SYSTEMS

Organized by
IIIS

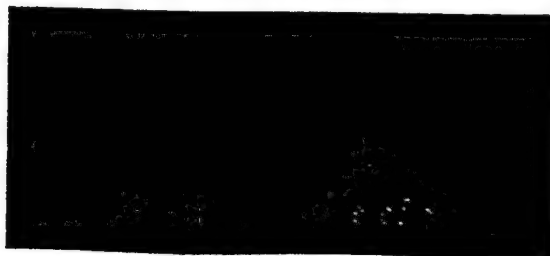


International
Institute of
Informatics and
Systemics

Member of the International
Federation of Systems Research

IFSR

Co-organized by IEEE Computer Society
(Chapter: Venezuela)



Spatial Data Mining Using Fuzzy Logic in an Object-Oriented Geographical Information Database

Maria A. Cobb

Department of Computer Science & Statistics
University of Southern Mississippi
Hattiesburg, MS 39406-5106, USA

Miyi Chung, Ruth Wilson, Kevin Shaw
Naval Research Laboratory
StennisSpace Center, MS 39529, USA

Frederick E. Petry

Center for Intelligent and Knowledge Based Systems
Department of Electrical Engineering & Computer Science
Tulane University
New Orleans, LA 70118, USA

ABSTRACT

The Mapping Sciences Section of the Naval Research Laboratory, Stennis Space Center, has realized the enormous benefits of spatial data warehousing and database integration with the implementation of the Geospatial Information Database (GIDB). An object-oriented approach was used to develop an object model that could be easily expanded to include all geographic data types. With the base of object-oriented technology, standards such as Common Object Request Broker Architecture (CORBA) and Virtual Reality Modeling Language (VRML) enabled 2-dimensional as well as 3-dimensional display over the internet.

However, in the process of developing the GIDB system, the question of what to do with all the data became an inevitable question. Data exist to be used and exploited by users, but what can users do with all the data? Is the availability of so much information overwhelming to the users? The use of spatial data mining techniques to help users make sense of the wealth of data in the GIDB is the focus of this paper. After general discussions of the topic of spatial data mining, we then present a specific technique for integrating a fuzzy set model for spatial relationship determination with the object-oriented model of the GIDB.

Keywords: spatial data mining, object-oriented, fuzzy spatial relationships.

1. INTRODUCTION

The Naval Research Laboratory with the University of Southern Mississippi, Tulane University, and Planning Systems Incorporated, has developed a Geospatial Information Database (GIDB) which allows the combination of National Imagery and Mapping Agency (NIMA) data (multiple data types, e.g., raster, vector, text) into a single integrated object-oriented database. The GIDB is also Common Object Request Broker Architecture (CORBA)-compliant and directly accessible via the Internet with support for 3D rendering. The GIDB allows

storage of complex data types (e.g., video and audio) with the traditional NIMA data types (raster, vector, and text) to enable area-of-interest queries as well as constrained queries (e.g., display all buildings within 30 meters of the plaza area).

The storage of multiple, complex spatial data types implies a wealth of available information to GIDB users. The GIDB supports both simple and advanced queries related to both spatial (position, geometry, topology, etc.) and non-spatial (attribute) properties of the data. However, improvements to these described query capabilities are currently underway in the form of the addition of a spatial data mining component. Spatial data mining is the determination of spatial or attribute information not explicitly stored in the database. Various techniques exist for "mining" of the data. The basics of it involve determining, through inference and other logical methods, relationships among the data.

Spatial data mining is a highly desirable addition to the GIDB for two reasons. First, the sheer volume of the data and the complexity of the data types is overwhelming to new or naive users. Spatial data mining techniques can be used to present a tailored, simplified view or schema of the data. The simplified schema can then be used to help users navigate the database. Second, spatial data mining can be used to aid users involved in advanced analysis of the data, and in high-level querying. In this case, non-obvious, implicit relationships among the data and even among the data types can be determined and made available for querying from the user.

Presently, the focus of our work is on the application of spatial data mining for advanced spatial and non-spatial queries. Non-spatial, or attribute queries, are relevant to two-thirds of the "families" of data contained in the GIDB—vector and text. Vector data is the most complex of the data types, consisting of geometric (positional) data and attribute (non-spatial) data, as well as topological relationships. Examples of attribute data include the width of a road, the purpose/use of a building, or the number of lanes of a highway. Examples of topological data include the designation of two roads that intersect, or two area features that overlap. Text data is stored in SGML format,

providing hypertext capabilities directly from the database. Text paragraphs are associated with positional characteristics in the form of latitudinal and longitudinal coordinates, thus giving the textual data a spatial component. Text is used for indicating, for example, sailing directions for a specific area, or dangerous marine animals. Multimedia data in the form of video and audio clips and images are used in conjunction with the text to provide the user with complete information.

Spatial information, on the other hand, takes the form of vector coordinates for the vector data, and image frames and subframes for the raster data. As mentioned above, text data also has a spatial component in the form of an indexing coordinate, similar to that used in gazetteers.

The examples of data types given above should convey the complexity of spatial and non-spatial data available in the GIDB. All data is stored within an internet-accessible object-oriented database. The object-oriented database schema is used as the basis for data mining of information related to metadata-level concepts, while the data instantiations are used for determining specific value-related data mining queries.

Based on the GIDB developed by NRL Code 7441, this research is exploring spatial reasoning within a spatial and object-oriented domain. Each of the fields is still pre-mature although more work has been published on spatial reasoning in recent years. An effort to incorporate research in spatial data mining and exploring characteristics of object-oriented technology for spatial data is the research domain of this paper. In particular, we focus on the application of fuzzy logic techniques for implementing spatial data mining for topological and directional relationships.

The following section provides background information on the motivation for spatial data warehousing/mining needs for military users. Section 3 provides an overview of spatial data mining followed by current research on data mining in object-oriented databases in Section 4. Since, this paper will use GIDB as a baseline application for spatial and object-oriented data mining, an overview of the application is also provided. Section 5 presents a framework for fuzzy modeling of spatial relationships. Section 6 continues with the integration of this model with the GIDB application. Section 7 explores the impact of spatial access methods on the model, and section 8 concludes with the potential growth of this field, as well as needed research topics in the area of spatial data mining in object-oriented databases.

2. BACKGROUND

With the explosion in the amount and availability of digital geographic data, many users are frustrated in trying to collect, compile, and analyze data in different formats (e.g., textbook, a chart, vector data) and over different display means (e.g., different windows for different visualization). Users want to collect and display the resulting information for easier compilation and analysis. Military digital mapping have these same needs.

Military users are dependent on maps to plan and conduct their operations. To military users, maps or mapping data capture the real-world entities that allow them to execute mission planning, mission maneuvering, and tactical operation planning. The

National Imagery and Mapping Agency (NIMA) provides map and mapping data to military users. NIMA currently has three digital mapping data formats: vector data in Vector Product Format (VPF), raster data in Raster Product Format (RPF), and text data in Text Product Standard (TPS). When military users, such as marines, have a need for all mapping information available over a certain geographic region or an area of interest (AOI), they ask NIMA to provide such information.

Once marines receive the data from NIMA, they are faced with more difficulties. Necessary information may be in VPF, RPF, and TPS format. It would make sense to have a system that would geo-reference the data from different data formats and display. Yet, the military does not currently have a system that could display the three data types together. Furthermore, there is not a system that could digitally catalog the data for easier and faster access later.

This predicament can be analyzed as having the following components: *data*, *integration*, and *visualization* (figure 1). The data component deals with the differences in the data format. Integration addresses the issue of data access across different data formats. Finally, visualization is the component in which the information content of the integrated data is displayed. With these concepts, terms such as *data warehouse* for storing data, *integrated database* for allowing seamless data access across different data formats, and *database management systems* for easier and faster access have become popular phrases among data users.

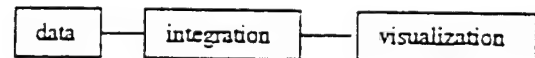


Figure 1. Target areas for improving users' interface with spatial data.

Next, the question becomes how to best utilize the integrated data, otherwise known as *data analysis*. Common data analysis efforts involved one or more analysts who became experts on the data, providing summaries and generating reports. Stored data provide useful information; however, raw data that has had no interpretation applied is rarely of direct benefit, especially in an integrated environment. Thus, we must add another component to the system (figure 2).

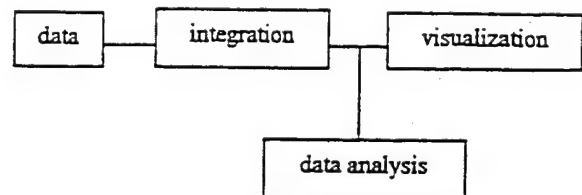


Figure 2. New target area for improvement, data analysis, integrated with prior system.

The true value of data is predicated on the ability to extract useful information for decision support or exploration, and on understanding the phenomena governing the data source. In an

effort to best analyze the integrated data, the following must be considered:

A wealth of stored information lies within the integrated data.

Potentially more information could be gleaned or inferred from the stored data.

Possible relationships and associations that exist between and among data could provide information that could not be found from the stored information.

Stored data, inferred information, and information from observing relationships and associations between and among data meet one objective—finding or discovering more information or knowledge.

Data mining assumes the role of advanced query on data. Relevant data are first examined. Then, various predicates are imposed on the queried objects to infer and compile the meaning of the collectively queried data. In summary, data mining is a query process that explores distinct information, as well as intrinsic relationships and associations between and among data.

3. SPATIAL DATA MINING OVERVIEW

When a user is given a set of data and a task to perform, the data is compiled, interpreted, and reasoned within the context of accomplishing the task. It is unrealistic for users to examine a large volume of spatial data in detail and extract interesting knowledge or general characteristics from spatial databases. Spatial data mining is a field of study that mines knowledge from large amounts of spatial data. Discovering knowledge or data mining in spatial databases is the extraction of interesting spatial patterns and features, general relationships between spatial and non-spatial data, and other general data characteristics not explicitly stored in spatial databases [15]. Such discovery may play an important role in understanding spatial data, capturing intrinsic relationships between spatial and non-spatial data, presenting data regularity in a concise manner, and reorganizing spatial databases to accommodate data semantics and achieve high performance [14].

Geographic, or mapping, data consist of spatial descriptions and non-spatial descriptions. Each representation of a real-world geographic entity is known as a *feature*. Spatial descriptions of features provide geometrical information, such as spatial location, perimeter (boundary) and area, and topological information such as adjacency, inclusion, etc. Non-spatial descriptions of spatial data are considered *attributes*. Attributes can include information such as a feature's name, and ancillary information useful for situation analysis. Data mining, therefore, consists of using these two classes of data to infer and discover more information that was not explicitly stored.

Much study has been devoted to spatial data mining. Prototypes such as GeoMiner [12] provide a spatial data mining concept. Koperski describes three primitives of spatial data mining:

Spatial characteristic rule- general description of spatial data.

Spatial discriminant rule- general description of the features discriminating or contrasting a class of spatial data from other class(es).

Spatial association rule- rules which describe the implication of one or a set of features by another set.

Through data mining, one or a combination of the three rules can be discovered. Stored information may only provide finite and explicit characteristics of spatial data. However, data mining may provide other characteristics of certain spatial data based on other spatial data. A spatial discriminant rule provides those characteristics that are distinct from other spatial data, e.g., roads along a coast tend to be two-lane roads. Spatial association rule provides more compound description of spatial data when certain relationships and associations can be made to other spatial data, e.g., buildings on highly elevated areas have pitched roofs.

Several algorithms for knowledge discovery are used based on each or a combination of the three rules. These may consist of, for example: generalization, clustering, exploring spatial associations, and using approximation and aggregation. Those algorithms that discover spatial characteristic rules are considered within the classes of generalization and clustering. Generalization requires extensive background knowledge. A *concept hierarchy* is used to allow spatial features to be generalized (bottom-up approach), or specified (top-down approach). For spatial data, two types of concept hierarchies are needed, spatial and non-spatial or attribute-based. A spatial concept hierarchy can be viewed as a breakdown of spatial entities by earth, continents, countries, providences, states, cities, etc. This can be achieved by utilizing an appropriate spatial indexing scheme, e.g., quadtree, r-tree. An exhaustive study on different spatial indexing schemes is presented in [11, 17].

Koperski listed three possible approaches for an attribute-based concept hierarchy: (1) climb the concept hierarchy when attribute values are changed to a general value, (2) remove attributes when further generalization is impossible or too many different values exist for an attribute, or (3) merge identical attribute values [14].

Spatial data mining involving clustering eliminates the need for background knowledge. Based on the characteristics and nature of data, a representative object is searched while clustering those related objects together. Several approaches are presented in Koperski's paper: PAM, CLARA, CLARANS, CF trees, and BIRCH [14]. All these approaches use attributes to characterize each feature by its similarities among other features.

Spatial association rules discover relationships among spatial objects. Spatial association rule discovery by Koperski and Han [15] requires a minimum support or minimum confidence. This constrains the search and discovery to be over those areas that at least meet this minimum support or confidence level. Therefore, a spatial association rule is of the form, $X \rightarrow Y (c\%)$, where X and Y are spatial or non-spatial predicates and $c\%$ is the confidence of the rule. Spatial predicates may be derived using topological relationships, e.g., *intersects*, *close_by*, *adjacent_to*.

4. OBJECT-ORIENTED DATABASE MINING

An object-oriented data model provides rich data structure and semantics more relevant to complex data, such as spatial data. The paradigm has characteristics such as class hierarchies, data and behavior, and polymorphism. These characteristics will augment or impact data mining within an object-oriented data domain. For example, class hierarchies are rigid and not very flexible; schema migration and evolution could potentially impact an existing data mining model within a given system. As always, a careful design of the data model is required before a system is implemented.

Class hierarchy could support a concept hierarchy if class definitions are deep. Property inheritance is assumed within class hierarchy consideration. Class membership is determined by the similarities that are inherent in all sub-classes. Thus, object model and class definition need to be considered seriously before developing an object-oriented data model for data mining.

Derivation of a concept hierarchy from a class hierarchy provides information about objects from general to specific or vice-versa. Thus, a class hierarchy tree may be used as a means of generalizing an object to its super- or parent class object. Objects can be clustered based on the class membership. Han [12] provides three characteristics of each object in a class pertaining to data mining:

- An object identifier,
- A set of attributes, and
- A set of methods that specify the computational routines or rules associated with the object class.

An object identifier may be an index to a class in a class hierarchy. Indexing schemes are used to facilitate quick data access in a large database. Much work on indexing schemes in object-oriented data models have been published, e.g., Class Hierarchy index (CH-index), nested-index, and multi-index [2]. An organization of classes by some indexing scheme corresponds to a class hierarchical structure. Thus, a class definition can be generalized from sub-class to super class via the indexing scheme or vice-versa.

Updating the indexing scheme based on schema evolution becomes an issue. Han implies that the object identifier needs to remain unchanged over structural reorganization of data. However, schema evolution does take place unless careful design with all parameters known during the design stage is made.

A state of an object is determined by its attributes. A set of attributes can be as simple as an integer type, or can be as complicated as a reference to another object. Within an object-oriented data model, there may exist *composite objects*, that is, objects that are composed of other objects, each of which may be composed of more objects, etc. The entire set of such relationships is known as a composition hierarchy, or web. These relationships greatly increase the complexity of an object definition. A depth of information traversal determines the level of data mining for complex objects. However, this is also an advantage for object-oriented data. By tapping into one object, all information or state of its information can be found and

inferred. Inference may result from an association or a relationship that exists between or among objects by reference. Simple traversal through an object web could cause the discovery of an association rule between and among objects

An object also encapsulates its behavior or functions. Certain functions could be implemented to support data mining. In reference to the data encapsulation advantage for complex objects, an object identifier could be found as a result of an attribute, or as a result of a function invoked to find that information. Certain behavior allows specific functions to be executed based on criteria imposed on the execution. This allows a dynamic and parametric invocation of some known behavior.

Object-oriented design and data structure can affect the non-spatial portion of the spatial data in data mining. A class hierarchy with concept hierarchy will support generalization of object descriptions. The state of an object could be readily accessed for a given object. An indexing scheme could be used to expedite the access. Once the state is known and found, an object could be clustered with similar objects. In addition, based on the references to other objects, an object composition hierarchy could provide a window to infer relationships and associations that exist between and among objects. Based on parametric invocation of behavior, an object can be constrained to behave a certain way under certain conditions to provide varying knowledge about an object.

5. FUZZY MODEL FOR SPATIAL RELATIONSHIP MINING

The ability to distinguish between similar spatial relationships and to communicate subtle differences in such relationships is a difficult task for automated systems. An especially difficult task is to determine the directional relationship between two-dimensional features. Fuzzy methods associated with linguistic variables [19] is the most promising approach so far for the problem of spatial relationship determination.

As an example, consider the three scenes pictured in figure 3. In figure 3(a) it is unclear whether the statement "A is west of B" or "A is southwest of B" better describes the directional relationship between the two. In figure 3(b), however, it is much less controversial to state simply that "A is west of B." Similarly, in figure 1(c), most would agree that now "A is southwest of B." When given a choice, however, many people would choose to include "hedges" in an attempt to convey more accurately the pictured relationship. For example, in describing figure (b), one might state that "A is *mostly* west of B." A similar problem occurs in the description of topological relationships. Because human reasoning is typically qualitatively, rather than quantitatively, based, people do not often care to know, for example, that "86% of object A overlaps 4% of object B." It is more useful to simply state that "Most of object A overlaps little of object B."

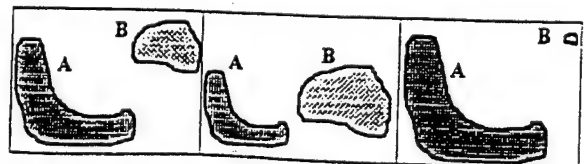


Figure 3. Example of the effect of size on directional relationship determination.

Previous work [5-8] has been performed in the arena of the definition and interpretation of fuzzy binary spatial relationships. This is used as a basis of our spatial data mining research. In particular, we concentrate on a data structure that represents topological and directional relationships, as well as supplementary information needed for fuzzy query processing. The data structure, known as an *abstract spatial graph* (ASG), represents a transformation of 2-dimensional space (areas) into 0-dimensional space (points). A complete set of ASGs for the original relationships, including a graphical representation and specific property sets, was developed in [7].

First-level topological relationship definitions are based on an extension of Allen's temporal relations [1] to the spatial domain. In this work, Allen showed that the seven relationships *before*, *meets*, *overlaps*, *starts*, *during*, *finishes* and *equal*, along with their inverses, hold as the complete set of relationships between two intervals. Cobb [7] extended these to two dimensions by defining a spatial relationship as a tuple $[r_x, r_y]$, where r_x is the one of Allen's relationships that represents the spatial relationship between two objects in the x direction, and r_y is likewise defined for the y direction. Objects involved are assumed to be enclosed by Minimum Bounding Rectangles (MBRs). (VPF, along with many other spatial data formats, provides MBRs as approximate feature boundaries in conjunction with detailed boundary representations.)

The construction of an ASG for a binary spatial relationship is dependent upon these object sub-groups. Each object sub-group is represented as a node on the ASG. Pictorially, ASG's are represented in a polar graph notation, where different node representations are used to distinguish between the objects involved in the relationship. The origin node represents the reference area of the relationship, which can be a sub-group of one of the objects, an overlapping area, or a common boundary. An example of an ASG and its corresponding relationship is shown in figure 2.

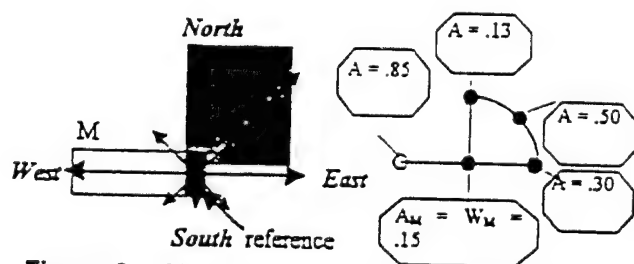


Figure 2. M [overlaps, starts] N relationship and ASG.

Each node in an ASG has associated area weights and total node weights to provide support for fuzzy query processing. These weights provide information concerning the degree of participation in a relationship and relative direction, respectively, and are used to define fuzzy qualifiers for the query language. Specifically, the weights are intended to support queries of the nature "To what degree is region A south of region B?", or "How much of region A overlaps region B (qualitatively speaking) ?". The exact manner in which the weights are computed can be found in [5, 7], while a discussion of variations on the weight calculations based on MBR refinements is the focus of [9].

By assigning ranges of area weights to linguistic terms we can provide a basis for processing queries concerning qualitatively defined relationships. The set given below is one example of how this may be done.

{all (96-100%), most (60-95%), some (30-59%), little (6-29%), none (0-5%)}

Node weights are utilized in a similar manner to provide qualitative directional relationship information. The purpose of node weights is to answer the *extent* to which an object can be considered at a given direction in relation to another object. Again, ranges are provided that define a linguistic set useful for query purposes. These are given below.

{directly (96-100%), mostly (60-95%), slightly (30-59%), somewhat (6-29%), not (0-5%)}

The use of these qualifiers is illustrated in the following:

- Is object B somewhat north of object A?
- Retrieve an object directly west of object A.
- Does most of object A overlap some of object B?

6. INTEGRATION OF THE ASG AND OO DATA MODELS

The ASG model provides a structured framework upon which fuzzy spatial queries related to topology and direction can be resolved. The full ASG model includes not only the spatial data structure described in the previous section, but also higher-level data structures suitable for building an index for such fuzzy queries, and a set of properties that can be exploited to resolve the queries efficiently. For example, a transitivity table for the complete set of relationships has been derived that could be used to quickly determine the 2D relationship between A and C, given the relationships between A and B, and between B and C.

The purpose of this section is to show how the ASG model can be incorporated into the existing GIDB OO data model. We concentrate on two aspects: (1) integration at the level of the geographic features, and (2) integration at the indexing level. The first step is to show how the relationships can be represented in the GIDB OO model. We then discuss, in general, how an indexing scheme can be merged with the current quadtree implementation for query purposes.

Every VPF feature, both in its original relational, and in its transformed object format, has an MBR represented through a pair of coordinates, one for the lower left corner, and one for the upper right corner of the bounding box. Because the 2D relationships upon which the ASGs are formed are well-defined, based on tri-valued ($<$, $>$, $=$) relations of the bounding box coordinates, we can store this set of relationship definitions as a global object. Indexing of these definitions based on the mutually exclusive relationship sets derived in [diss] (i.e., surrounded-by, tangent, etc.) can be performed to quickly derive a fuzzy relationship label.

For example, the inexact relationship *partially-surrounded-by* is mapped to a set of 7 basic relationships defined by MBR bounding box interactions. We know that any one of those 7

binary relationships will be equated to the *partially-surrounded-by* spatial relationship. At that point, the ASG area weights are used to impart qualitative (linguistic) refinements to the relationship. Figure 5 shows how a cross-indexing scheme can be maintained to provide efficient access for both location and relationship-based queries. The figure shows the pointers (implemented as instance variables) from the features to a corresponding ASG. Obviously, a feature has a one-to-many relationship to ASGs, while each ASG corresponds to exactly two features. The feature, represented strictly as a pointer to an object, is represented in the quadtree structure to facilitate efficient location-based querying, and in the ASG collection (indexed by fuzzy relationship terms, as mentioned earlier) for potential relationship data mining operations.

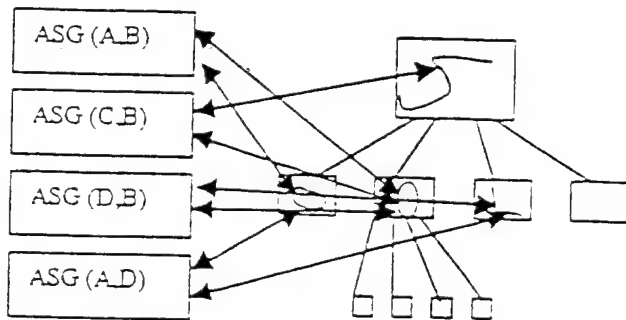


Figure 5. Quadtree and ASG indexing for queries.

Because the number of possible binary relationships in a database with N objects is $O(N^2)$ it is not very likely that one would want to calculate and store all the relationships at one time. It is more practical in the cases of large spatial databases to compute the relationships as needed based on queries (e.g., find all adjacent building structures) and to store any computed relationships for help in determining subsequent relationships. This could be accomplished, for example, through the transitivity table mentioned earlier, or through MBR filtering techniques such as that described in Clementini [3].

7. SPATIAL ACCESS METHOD IMPLICATIONS

Various models for data mining have been proposed such as DBLEARN/DBMINER [12], Holsheimer et al., [13] and Matheus [16]. A general architecture proposed by Matheus is used in this paper as shown in figure 6. This section will focus on the *DB Interface* component of the knowledge discovery process.

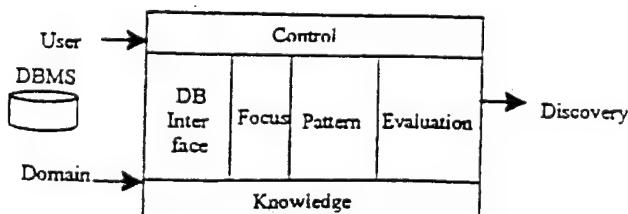


Figure 6. General data mining architecture by Matheus [16].

In general, databases store and maintain persistent data. To support any application, the data would have to be fetched from the database as a form of a query request. Formally, a purpose of an interface to a database is to facilitate a selection of set of objects that meet user defined constraints. Shekar et.al. lists several interfaces to DBMS such as index structure, spatial join, and views [18].

Indexing structures support faster access and retrieval, thereby providing an efficient processing of object fetch. Koperski [14] stated that the *DB Interface* in general has spatial data structures as well as query optimization. Ester [10] indicated that a spatial access mechanism (SAM) provides efficient processing of a selection of objects that fulfill some conditions specified by a user from the database. For spatial data, R^* tree has been gaining research interest [10, 11, 14]. However, Gaede [11] reminds the spatial data community that no one SAM is superior from on over another. Instead, he states, "Both time and space efficiency of an access method strongly depend on the data processed and the queries asked."

The GIDB uses a quadtree as the interface to the object-oriented DBMS. Quadtree is an approximation mechanism based on a MBRs. It is a simple and intuitive method of organizing spatial data in a geographically hierarchical or geographically clustering manner. The principle behind the construction of a quadtree is based on a recursive division of regions into four equal-sized cells, quadrants, until the cell that will minimally contain the MBR is found. For any MBR that overlaps more than one quadrant, the MBR is maintained at the parent quadrant.

A SAM, such as quadtree, along with object-orientation provides advanced search and analysis capabilities. Properties such as data encapsulation and object nesting allow spatial search as well as attribute search to take place. As a spatial search is conducted, spatial relationships can be computed and determined among objects in a quadtree. As indicated, a quadtree is a geographically hierarchical tree. In other words, objects are clustered spatially. Objects that are geographically located in the northwest corner of the United States, for example, will be placed along one branch of a quadtree. Therefore, retrieval will only occur along one branch of a tree for data in the same region.

In GIDB, a quadtree implementation has the following object structures,

VPFSpatialDataManager *manager of any SAM
(For GIDB, a quadtree is used.)

topCell * root of a tree

maxCell * quadtree creation is based on maximum level of trees

storedContainer * an instance of VPFSpatialContainer

VPFSpatialContainer

attribDict * a collection keyed by Feature and Attribute Coding Catalogue
features * a collection of features that can be minimally contained in the cell
cell * a container can have at most four equal-sized cells
asg * a collection of asg indices between features

VPFSpatialDataCell * class definition of each quadrant of a quadtree

superCell * a backpointer to the parent cell
level * current level of a cell
manager * a backpointer to VPFSpatialDataManager
origin * lower left corner of a quadrant
corner * top left corner of a quadrant
lowerlevelWidth * width of a cell in next level of a quadtree
container * a reference to an instance of VPFSpatialContainer
id * unique identification of a cell in a quadtree

VPFSpatialDataManager and VPFSpatialDataCell classes describe the quadtree structure. VPFSpatialContainer class actually contains the data. Each instance of VPFSpatialContainer class contains a collection of *features*, *attribDict*, and *asg*. *Features* is a collection of features whose boundingBox is minimally contained by a cell. An *attribDict* contains an attribute indexing scheme based on the attributes of features in the *features* collection. Due to the data encapsulation property of OO, each feature knows all its attribute information; however, an indexing scheme is used for performance reasons. The *attribDict* is indexed by the attribute type. For each attribute type, another collection is maintained if an attribute is a multivalued attribute. Otherwise, a collection of those features that contain such an attribute will be collected in *attribDict*. For a multi-valued attribute, a collection is maintained that uses the actual value of the attribute as the index. For each attribute value, a collection of features that has the attribute and the attribute value is maintained. An *asg* is a collection of asg values between feature A and feature B, as described in the previous section.

If a user poses a query such as "find all roads that have two lanes that are close to government buildings," two types of search need to be in progress: attribute-constrained search as well as spatial-constrained search. In all spatial searches, an assumption is made that an area-of-interest (AOI) has been selected. Based on the AOI, the search begins at the root of a quadtree and progresses until all cells that intersect the AOI are selected.

First, an attribute-constrained search process is described. As the traversal along a quadtree continues, an *attribDict* is used to find all roads and government buildings. NIMA's VPF uses Feature Attribute Coding Catalog (FACC) that indicates whether a feature is a road (AP010, AP020, AP030) or a government building (AH010, AH020, AH050, AH060, AH070). Each instance of road also maintains an attribute LTN

which provides the track or lane number. So, as the traversal of the quadtree is in progress, those cells that have an entry for LTN attribute as well as any FACC of government buildings will be collected into a collection of results.

The spatial search constraint is the user request to find features that are *close to* feature B. In order to build asg indices for each feature combination, a topological relationship among features that meet the attribute constraints must be evaluated. In other words, a spatial join among all road and government building features in the AOI must be computed to determine if any features are adjacent. Once the features are determined to be adjacent then an asg value is computed. Thus, a spatial search along a quadtree must first compute topological relationship among features. Then those features that meet the topological relationship will be used to compute asg values among the features.

8. CONCLUDING REMARKS

Clearly, spatial data mining techniques are needed to enhance usability of the massive amounts of currently available spatial data. The complexity of spatial data, with inherent composite relationships, and spatial and non-spatial attributes, warrants some way in which to both provide a simplified meta-level view of data to users, as well as automatic discovery of rules and relationships among the data. Object-oriented modeling techniques are the accepted paradigm for representing spatial data, and data mining practices fit well within this framework. Many of the object-oriented principles, such as inheritance and composition, can be exploited within a data mining context to provide powerful inferencing mechanisms.

In this paper, we first overviewed the principles of spatial data mining within an object-oriented framework. It was then shown how an existing fuzzy spatial relationship model could be integrated with an object-oriented spatial data schema to provide relationship definitions for high-level fuzzy querying of such relationships by the users. This is the initial step in developing a spatial data mining framework based on abstract spatial graphs.

Although some work in the integration of data mining and object-oriented modeling has been performed, the field is certainly not mature as of this time. More work in formal methods of integration is warranted, as is research in data mining issues specific to spatial data. Our plans for future work include an implementation of the model described in section 5, as well as the development of a full-featured non-spatial attribute data mining component.

9. ACKNOWLEDGEMENTS

We would like to acknowledge Marine Corps Warfighting Laboratory (MCWL), Program Element number 0603640M, for funding this project, with special thanks to Lt. Col. Bott and Lt. Col. Durham as program managers of the IMMACCS project.

10. REFERENCES

- [1] James F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, Vol. 26, No. 11, November 1983, pp. 832-843.
- [2] Elisa Bertino, Beng Chin Ooi, Ron Sacks-Davis, Kiam-Lee Tan, Justin Zobel, Boris Shidlovsky, and Barbara Cantania, *Indexing Techniques for Advanced Database Systems*. Kluwer Academic Publishers, 1997, pp. 1-38.
- [3] E. Clementini, J. Sharma and M.J. Egenhofer, "Modelling Topological and Spatial Relations: Strategies Needs," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 1, 1994, pp. 45-55.
- [4] "for Query Processing", *Computers and Graphics*, 18:6, 815-22.
- [5] M. Cobb and F. Petry, "Modeling Spatial Relationships within a Fuzzy Framework," *Journal of the American Society for Information Science*, Vol. 49, No. 3, 1998, pp. 253-266.
- [6] M. Cobb and F. Petry, "Integration of a Fuzzy Query Framework with Existing Spatial Query Languages," *Proc. Fifth IEEE Intl. Conf. On Fuzzy Systems (FUZZ-IEEE)*, New Orleans, LA, 1996, pp. 93-99.
- [7] M. Cobb, "An Approach for the Definition, Representation and Querying of Binary Topological and Directional Relationships between Two-Dimensional Objects," Ph.D. dissertation, Tulane University, New Orleans, LA, 1995.
- [8] M. Cobb and F. Petry, "Fuzzy Querying of Binary Relationships in Spatial Databases," *Proc. IEEE Intl. Conference on Systems, Man and Cybernetics*, Vancouver, BC, 1995, pp. 3624-3629.
- [9] M. Cobb and F. Petry, "Extensions to Geometric Approximations of Spatial Boundaries and Assessment of Fuzzy Spatial Relationships," *Proc. of the 1998 International Conference on Fuzzy Systems (FUZZ-IEEE '98)*, Anchorage, AK, May 4-9, 1998, pp. 220-225.
- [10] M. Ester, H. P. Kriegel, and X. Xu, "Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification," in *Proc. 4th International Symposium on Large Spatial Databases (SSD'95)*, Portland, Maine, 1995, pp. 67-82.
- [11] V. Gaede and O. Gunther, "Multidimensional Access Methods," in *ACM Computing Survey*, Vol. 30, No. 2, 1998, pp. 170-231.
- [12] Jiawei Han, Shojiro Nishio, and Hiroyuki Kawano, "Knowledge Discovery in Object-Oriented and Active Databases," in F. Fuchi and T. Yokoi (eds.), *Knowledge Building and Knowledge Sharing*, Ohmsha, Ltd. and IOS Press, 1994, pp. 221-230.
- [13] M. Holsheimer and M. Kersten, "Architectural Support for Data Mining," in *CWI Technical Report CS-R9429*, Amsterdam, The Netherlands, 1994.
- [14] Krzysztof Koperski, Junas Adhikary, and Jiawei Han, "Spatial Data Mining: Progress and Challenges Survey Paper," *SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, June 1996.
- [15] Krzysztof Koperski and Jiawei Han, "Data mining Methods for the Analysis of Large Geographic Databases," *Proc. 10th Annual Conference on GIS*, Vancouver, Canada, March 1996.
- [16] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro, "Systems for Knowledge Databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 6, 1993, pp. 903-913.
- [17] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1990.
- [18] Shashi Shekhar, Ravada Siva and Liu Xuan, "Spatial Databases: Accomplishments and Research Needs," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 1, 1999, pp. 45-55.
- [19] L. Zadeh, "The Concept of a Linguistic Variable and its Application to Approximate Reasoning," *Information Sciences*, Vol. 8, No. 3, 1975, pp. 199-249.

July 25-28, 2001

Vancouver, BC

A Clips-Based Implementation for Querying Binary Spatial Relationships

Huiqing Yang, Maria A. Cobb
 Department of Computer Science & Statistics
 University of Southern Mississippi
 Hattiesburg, MS, USA 39401-5601
 hyan2@orca.st.usm.edu
 Maria.Cobb@usm.edu

Kevin B. Shaw
 Naval Research Laboratory
 Mapping, Charting & Geodesy
 Stennis Space Center, MS, USA 39529
 shaw@nrlssc.navy.mil

Abstract

The power of spatial queries for analysis and planning purposes in many different application fields has drawn significant attention within the GIS research field. The extraction of meaningful information from spatial data requires specialized data structures, query languages and query processing strategies.

This paper is primarily concerned with the binary data structures that support the fuzzy queries of spatial relationships in two dimensions. For implementation purpose, the topological relations in this model are refined from a previously defined model. This modified binary spatial model will reduce the burden of geometric computation. Based on the modified binary spatial model, a CLIPS implementation for querying binary spatial relationships is investigated. Details about the query processing strategies are also provided.

1. Introduction

Geographic Information Systems (GIS) is an integrated technology that incorporates concept from computer graphics, spatial modeling and database management. The ability to perform queries on spatial data is essential to GIS and related systems. Due to the fact that the ability to extract information for query results is dependent on the underlying structure of data, a great deal of research efforts have focused on the modeling of spatial data. It is worth mentioning that the work in [1] provided a novel contribution to the problem of defining spatial relationships by considering inferences from topological and directional relations.

In earlier work [1], a spatial data model that represents binary topological and directional relationships between two 2-D objects was presented. A data structure called an Abstract Spatial Graph (ASG) was defined for the binary relationship that maintains all necessary information regarding topology and direction. For complete information on this model, we refer the reader to the cited references.

In this paper, we present an implementation of the modified structures based on the C Language Integrated Production System (CLIPS). CLIPS is a productive development and delivery expert system tool which provides a complete environment for the construction of rule and/or object-based expert systems [2, 3]. It is now maintained as public domain software. Because of its portability, extensibility, capabilities, and low-cost, CLIPS has received widespread acceptance throughout the government, industry and academia.

Based on the modified spatial relationship, rules are encoded using the public-domain CLIPS language. The CLIPS code is processed through the CLIPS expert systems engine to answer the topological and directional queries for binary spatial objects.

The paper is organized as follows. Section 2 describes the rules of spatial relations based on the binary spatial model, and investigates a data structure improvement for implementation purposes. Section 3 provides details about CLIPS programming strategies for query processing. The query result section follows, providing a sample of how the implementation works. Our conclusion and directions for further work are presented in section 5.

2. A Binary Spatial Model and Its Modification

For the purpose of the model, we first assume that objects involved can be enclosed in Minimum Bounding Rectangles (MBRs). Figure 1 shows two MBR objects in 2-dimensions, i.e., each object can be represented by a two-point abstraction that represents the lower-left and upper-right corners of the MBR.

A tuple $[r_x, r_y]$ represents the relationship between the objects in both the horizontal and vertical directions. Each of r_x and r_y is one of Allen's temporal relations [4] that represents the interaction of the objects in the x direction and y direction, respectively.

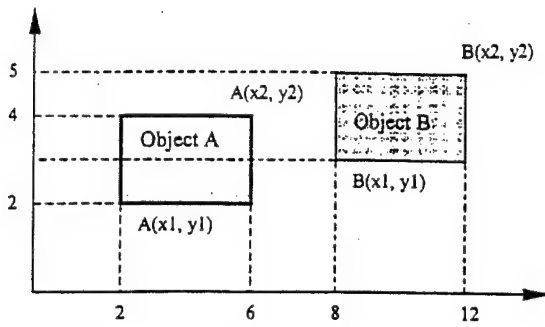


Figure 1. MBRs of two objects.

2.1 Basic Rule Sets of Binary Spatial Relations

Based on Allen's 13 temporal relations, a two-dimensional spatial data model was achieved. The result is that 85 possible relationships are deduced, which include 49 base relationships and 36 inverse relations. These relationships are then used to define topological and directional relationships.

In this paper, three rule sets are used to represent the basic structure of this model. Consider two objects:

$$A(Ax1, Ay1) (Ax2, Ay2) \\ B(Bx1, By1) (Bx2, By2)$$

Now, taking one-point from each object, that is,

$$(A1, A2) = (Ax1, Ax2) \text{ or } (Ay1, Ay2) \\ (B1, B2) = (Bx1, Bx2) \text{ or } (By1, By2),$$

we will present the implementation process.

Rule Set 1: Define a set of non-ambiguous relationships.

Consider one direction, the temporal relation between object A and object B can be defined as:

1. IF $\langle A2 < B1 \rangle$ THEN $\langle \text{before} \rangle$
IF $\langle B2 < A1 \rangle$ THEN $\langle \text{before}^{-1} \rangle$
2. IF $\langle A2 = B1 \rangle$ THEN $\langle \text{meet} \rangle$
IF $\langle B2 = A1 \rangle$ THEN $\langle \text{meet}^{-1} \rangle$
3. IF $\langle A1 < B1 < A2 < B2 \rangle$ THEN $\langle \text{overlap} \rangle$
IF $\langle B1 < A1 < B2 < A2 \rangle$ THEN $\langle \text{overlap}^{-1} \rangle$
4. IF $\langle B1 < A1 < A2 = B2 \rangle$ THEN $\langle \text{finish} \rangle$
IF $\langle A1 < B1 < A2 = B2 \rangle$ THEN $\langle \text{finish}^{-1} \rangle$
5. IF $\langle B1 < A1 < A2 < B2 \rangle$ THEN $\langle \text{during} \rangle$
IF $\langle A1 < B1 < B2 < A2 \rangle$ THEN $\langle \text{during}^{-1} \rangle$
6. IF $\langle A1 = B1 < A2 < B2 \rangle$ THEN $\langle \text{start} \rangle$
IF $\langle B1 = A1 < B2 < A2 \rangle$ THEN $\langle \text{start}^{-1} \rangle$
7. IF $\langle A1 = B1 < A2 = B2 \rangle$ THEN $\langle \text{equal} \rangle$

Figure 2. Defining a set of non-ambiguous relations.

Simply, this rule set can be expressed as:

$$r_x = (b, m, o, f, d, s, =, b', m', o', f', s'),$$

where each relationship and its inverse is represented by its initial letter, e.g., 'b' \rightarrow 'before.'

In the y-direction, the same rules can be applied. Moreover, there are two additional rules that apply:

1. $A(r_x^{-1}, r_y)B = B(r_x, r_y^{-1})A$
2. $A(r_x^{-1}, r_y^{-1})B = B(r_x, r_y)A$

Rule Set 2: Define a set of topological relationships.

Based on the eighty-five basic relationships, the topological relation set can be defined as:

$T = \{\text{disjoint, tangent, surrounded-by, partially-surrounded, surrounded-by, partially-surrounds, overlapped-by, overlaps, x-subspace, y-subspace, y-subspace-by}\}$

Figure 3 shows a subset of the rules for topological relationships. [1] provides greater details on this.

```
IF <dd> THEN <A surrounded-by B>
IF <oo'|os'|of'> THEN <A overlapped-by B>
IF <s|=d|=f|=|=o> THEN <A x-subspace B>
IF <=s|=d|=f|=|=o> THEN <A y-subspace B>
```

Figure 3. Topological relationship rule set.

Rule Set 3: Define the set of directional relationships.

Directional relationships are heavily used in everyday life. The most commonly used are the cardinal directions and their refinements. In the same way as previously seen for topological relationships, the directional set can be defined as:

$$D = \{\text{North, East, South, West, North-East, South-East, South-West, North-West}\}$$

Figure 4 shows two of the rules for directions.

```
IF <dd|df|fd|do|ds|ff|d|=|fo|fs|f|=|dd'|do'|ds'|fd'|df'|fo'|fs'>
THEN <A East B>

IF <dd|do|ds|fo|fs|db|dm|fb|fm|dd'|fd'|df'|ff'>
THEN <A South East B>
```

Figure 4. Two directional relationship rules.

2.2 Define ASG for Fuzzy Querying

Three basic rule sets can support the basic binary spatial querying, i.e. the querying without specific degree information. Researchers [5-6] have shown that the directional relationships are fuzzy concepts since they depend on human interpretation. In addition to supplementary information needed for

fuzzy query processing, a data structure, known as an abstract spatial graph (ASG), was also presented in previous work [1]. The concept is based on the tasks of defining reference areas, partitioning MBR's into object sub-groups, and assigning each object sub-group to a node on the ASG.

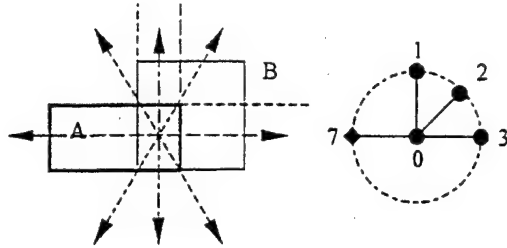


Figure 5. A [overlaps, start] B and corresponding ASG.

Figure 5 shows the geometry of the ASG, in which nodes 0-3 belong to object B and nodes 0 and 7 represent object A. Each node has associated weights that store fuzzy information.

2.3 Modifying AGS for CLIPS Implementation

The topological relations have been found useful for increasing the speed of spatial queries [5]. For implementation purpose, we analyze the geometric characteristics of topological relationships. Excepting the disjoint relation, all other relations have a similar geometry; that is, the reference area is part of both objects involved. Thus, the original topological relation set can be reduced or reclassified to a binary topological set:

$$T \rightarrow T' = \{\text{disjoint, connected}\}$$

This new topological relation set is used in the CLIPS implementation.

For convenience of implementation and further investigation, the ASG is modified by mapping topological relationships to 9 nodes for both objects. Figure 6 represents the new ASG. Similarly, each node has associated weights. But differently, the weight in some node can be null depending on the different topological relations. In this new data structure, because each object is associated with its 9 nodes, it is not necessary to keep information related to whether a node belongs to object A or object B in the implementation. Furthermore, it is a flexible structure for fuzzy querying.

3. A CLIPS Implementation

In this section we show how CLIPS can be used to implement a the binary spatial relationships given

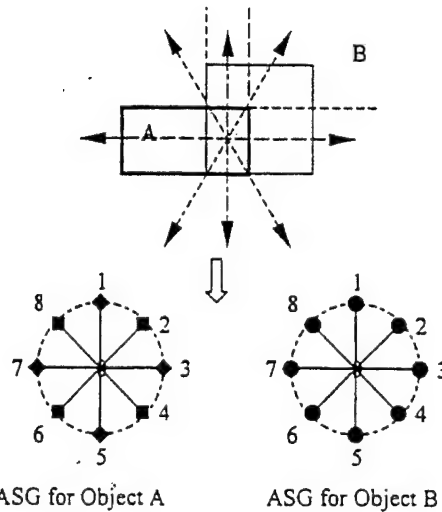


Figure 6. A [overlaps, start] B and new ASG

earlier. Considering the amount of computation involved in implementation, we take advantage of the *deffunction* construct that allows the addition of new functions without having to recompile and relink CLIPS. Several user-defined functions are written by using the CLIPS *deffunction* construct, which can be executed by CLIPS interpretively.

As a rule-based shell, CLIPS stores the knowledge in rules, which are logic-based structures. In the implementation, the basic three rules are defined by using *defrule* constructs. They provide the basic spatial information such as, *Object A is disjoint from Object B*, or *Object A is West of Object B*. For fuzzy querying purposes, extra functions and rules are defined that will support fuzzy querying.

The implementation is directly dependent upon the reduced topological relation set and modified ASG mentioned above.

3.1 Store All Facts in CLIPS

The facts are the critical resources for the querying. All details for binary spatial relations are contained in *deftemplate* facts. The type of information stored in the database includes the positions of two objects, the reference object, non-ambiguous relations, and topological relationship and directional relationships. Figure 7 shows the information stored in facts using CLIPS syntax. The corresponding data structures are declared by using *deftemplate* syntax.

```

(object-position (objectname A)
  (x1 2 ) (y1 2 ) (x2 3 ) (y2 5 ) )

(2D-relation (object1 A)
  (relations bd) (object2 B))

(topological-relationship (object1 A)
  (t_relation disjoint) (object2 B) )

(directional-relationship (object1 A)
  (d_relation West) (object2 B) )

(nodes (objectname A)
  (Central area_weight)
  (N area_weight direction_weight)
  :
  (NW area_weight direction_weight)
)

```

Figure 7. Facts stored in database.

3.2 Representing 2-D Relation in CLIPS

To represent 2-D temporal relations extended from Allen's relations, the *deffunction* construct in CLIPS is utilized. With this construct, a new function that implements Allen's relations in 1-D is defined directly in CLIPS. Figure 8 shows the *deffunction* for Allen's internal relations. The knowledge of the rules implemented in step one that define a set of non-ambiguous relationships is built by the *defrule* construct shown in Figure 9.

```

(deffunction AllenRelation
  (?A1 ?A2 ?B1 ?B2)
  (if (< ?A2 ?B1) then (bind ?relation b))
  (if (= ?A2 ?B1) then (bind ?relation m))
  (if (and (< ?A1 ?B1) (< ?B1 ?A2) (< ?A2 ?B2))
    then (bind ?relation o))
  (if (and (< ?B1 ?A1) (= ?A2 ?B2))
    then (bind ?relation f))
  (if (and (< ?B1 ?A1) (< ?A2 ?B2))
    then (bind ?relation d))
  (if (and (= ?B1 ?A1) (< ?A2 ?B2))
    then (bind ?relation s))
  (if (and (= ?B1 ?A1) (= ?A2 ?B2))
    then (bind ?relation =))
  return ?relation
)

```

Figure 8. *Deffunction* for Allen's interval relations.

The function `AllenRelation()` develops a set of temporal relations in 1-D. It accepts four arguments from a CLIPS program. When it is called, it returns the temporal relation that can be used in the rule for the application.

The *defrule* collects the relation facts in 2-D by calling `AllenRelation()`, and then puts the 2-D relation knowledge into facts.

```

(defrule define-2D-relation
  ?f3 <-(object-position (objectname ?A&A)
    (x1 ?Ax1) (y1 ?Ay1) (x2 ?Ax2) (y2
    ?Ay2))
  ?f4 <-(object-position (objectname ?B&B)
    (x1 ?Bx1) (y1 ?By1) (x2 ?Bx2) (y2
    ?By2))
  =>
  (bind ?x (AllenRelation ?Ax1 ?Ax2 ?Bx1
    ?Bx2))
  (bind ?y (AllenRelation ?Ay1 ?Ay2 ?By1
    ?By2))
  :
  (bind ?r (sym-cat ?x_relation
    ?y_relation))
  (assert (2D-relation (object1 ?A )
    (relations ?r) (object2 ?B ))))

```

Figure 9. *Defrule* to implement Rule Set 1.

3.3 Basic Binary Spatial Querying Using CLIPS

The basic queries are based on the primary topological set (Rule Set 2) and directional set (Rule Set 3). In this kind of querying, the degree to which one object lies in a particular direction with respect to a second object is not of concern. Figures 10 and 11 show CLIPS rule structures for topological relationship and directional relationship, respectively.

```

(defrule define-topological-relation
  (relationship (object1 ?A&A)
    (relations ?r) (object2 ?B&-A))
  =>
  (if (eq ?r dd)
    then (bind ?tr "is surrounded by"))
  :
  (if (numberp (member$ ?r
    (create$ oo' os' of' )))
    then (bind ?tr "is overlapped by"))
  (assert (topologic-relationship
    (object1 ?A) (t-relation ?tr)
    (object2 ?B) ) )

```

Figure 10. *Defrule* for topological relationship.

```

(defrule define-directional-relation
  (relationship (object1 ?A&A)
    (relations ?r) (object2 ?B&-A))
  =>
  (if (numberp (member$ ?r (create$ od of
    sd sf dd df fd ff =d =f
    ob' om' oo' os' ..... )))
    then (bind ?dr1 North))
  :
  (loop-for-count (?count 1 8) do
    (bind ?dr (nth$ ?count (create$ ?dr1
    ?dr2 ?dr3 ..... ?dr7 ?dr8)))
    (if (numberp (member$ ?dr (create$
    North East ..... West )))
      then
        (assert (directional-
        relationship
        (object1 ?A) (d-relation ?dr)
        (object2 ?B) ) ) ) )

```

Figure 11. *Defrule* for directional relationship.

3.4 Fuzzy Querying of Binary Spatial Relationships

Based on the new topological relation set and modified ASG data structure, we define three rules and four functions to support the processing of fuzzy queries. Query processing strategies are described as follows:

Step1. Find the reference area

Fuzzy variable *weights* store all fuzzy query information. In order to get weights for each node in the ASG, a reference area must first be found. Based on the fact that there are four points in the x-direction or y-direction, given two objects, a simplified approach to determine the reference area can be given.

Approach: The reference area is also treated as an MBR object. We take two middle points among the four points in each direction as the reference object position. It can be represented as $R = (R_{x1}, R_{y1}) (R_{x2}, R_{y2})$.

get-reference-object Rule and reference Function

Given two objects, the *get-reference-object* rule calls *reference* function to get the reference object position. The *reference* function accepts eight arguments that represent positions of two MBR objects, and finds the position for the reference object. Finally, it places the position information into the corresponding *object-position* fact.

Step2. Calculate weights

Based on the binary topological relations, a general method developed for connected relations is shown in Figure 12.

```

N_area = (Rx2 - Rx1) (Oy2 - Ry2)
NE_area = (Ox2 - Rx2) (Oy2 - Ry2)
E_area = (Ox2 - Rx2) (Ry2 - Ry1)
SE_area = (Ox2 - Rx2) (Ry1 - Oy1)
S_area = (Rx2 - Rx1) (Ry1 - Oy1)
SW_area = (Rx1 - Ox1) (Ry2 - Oy1)
W_area = (Rx1 - Ox1) (Ry2 - Ry1)
NW_area = (Rx1 - Ox1) (Oy2 - Ry2)

```

Figure 12. Formulas for area weight calculation.

In the figure, R represents the reference object, and O represents the one of two objects investigated. By adding some constraints, the general method for connected relations can also be applied to disjoint relations.

get-weight Rule and weights Function

Given two objects and their reference object, the *weights* function maps the object sub-group into 9 nodes for each object, and calculates the area weights and node weights. The CLIPS program passes nine arguments to *weights* function, that is, one for object identifier, four for object position, and four for reference position. The function asserts area weights to the corresponding nodes for fuzzy querying. The basic *weights* function structure is shown in figure 13. A related rule that activates the *weights* function.

```

(deffunction weights (?object ?Ox1 ?Oy1
                     ?Ox2 ?Oy2 ?Rx1 ?Ry1 ?Rx2 ?Ry2)

  (bind ?Total_area (* (- ?Ox2 ?Ox1)
                       (- ?Oy2 ?Oy1)))
  (bind ?C_area (/ (* (- ?Rx2 ?Rx1)
                     (- ?Ry2 ?Ry1)) ?Total_area))
  (if (and (<= ?Ox1 ?Rx1) (>= ?Ox2 ?Rx2))
      then
        (bind ?N_area (/ (* (- ?Rx2 ?Rx1)
                          (- ?Oy2 ?Ry2)) ?Total_area))
      else
        (bind ?N_area 0); for disjoint case)

  ;get the node weight for north direction
  (if (> ?N_area 0)
      then (bind ?N_len(- ?Oy2/(+ ?Ry2
                                   ?Ry1)2)))
  )
  (if (< ?N_len 0) then (bind ?N_len 0))
  (if (> ?N_len ?Longest)
      then (bind ?Longest ?N_len) )

  (assert (nodes (objectname ?object)
                 (C ?C_area )
                 (N ?N_area (/(* ?N_area ?N_len)
                              ?Longest)))
          . . . . . )

```

Figure 13. Function to calculate weights.

Step3. Get qualifier to implement Fuzzy querying

To provide support for fuzzy query processing, the fuzzy variable *weights* is assigned to the corresponding linguistic terms qualifier. The *fuzzyTq* function defines the topological qualifiers that represent the linguistic terms for area weight. Similarly the *fuzzyDq* function defines the directional qualifiers that represent the linguistic terms for node weight.

The fuzzy set for topological qualifiers is:

{all (0.96 - 1), most (0.6 - 0.95), some (0.3 - 0.59)
little (0.06 - 0.29), none (0 - 0.05) }

The fuzzy set for directional qualifiers is:

{directly (0.96 - 1), mostly (0.6 - 0.95), somewhat
(0.3 - 0.59), slightly (0.06 - 0.29), not (0 - 0.05) }

The *fuzzy-query* rule in figure 15 provides the fuzzy querying information by calling *fuzzyTq* and *fuzzyDq* functions.

```
(defrule fuzzy-query
  ?f3 <- (nodes (objectname ?A&A)
    (C ?C_area )
    (N ?N_area ?N_len)
    (NW ?NW_area ?NW_len) )
  =>
  (if (neq ?A B ) then (bind ?obj B )
    (loop-for-count (?count 1 8) do
      (bind ?dir (nth$ ?count (create$ North
        ... North_West)))
      (bind ?area_w (nth$ ?count (create$
        ?N_area ?NE_area ?E_area
        ... ?SE_area ?NW_area)))
      (bind ?node_w (nth$ ?count (create$
        ?N_len ?NE_len ?E_len
        ... ?W_len ?NW_len)))
      (bind ?tq (fuzzyTq ?A ?area_w
        ?dir ?obj))
      (bind ?dq (fuzzyDq ?A ?node_w
        ?dir ?obj))
      (if (and (neq ?tq non) (neq ?dq non ))
        then
          (printout t "query information" crlf)
      ) ) )
```

Figure 15. Fuzzy query rule.

All of the CLIPS code is processed through the CLIPS expert systems engine to answer the topological and directional queries for binary spatial objects.

4. Query Results

Consider two objects:

object A (1, 1) (5, 3) and
object B (4, 1) (8, 7).

When the *define-2D-relation* rule is fired, calling *AllenRelation* (1 5 4 8) will return 'o', and the second calling of *AllenRelation* (1 3 1 7) will return 's.' Finally, the relation 'os' is added to the *temporal-relation* fact.

When the *define-topological-relation* rule is fired, the topological information 'Object A overlaps Object B' is displayed. When the *define-directional-relation* rule is fired, 'Object A is South Object B, Object A is South West of Object B, and Object A is West of Object B' are provided for directional relations. When the *reference* rule is fired, the reference object R(4, 1) (5, 3) is asserted into the fact database. While

the *get-weight* rule is firing, area weights and node weights are assigned into 9 nodes for each object.

Finally, the *fuzzy-query* rule fires, providing the following fuzzy querying information:

Most of Object A is West of Object B
Object A is mostly West of Object B
⇒ *Most of Object A is mostly West of Object B*

5. Conclusion and Directions for Further Works

In this paper, the capabilities of a binary spatial data model and a CLIPS tool to support fuzzy topological and directional queries have been shown. The results demonstrate that CLIPS is a flexible, powerful, and intuitive tool that can be successfully applied to spatial database analysis.

Because the querying involves handling concepts expressed by verbal language, such as direction, area weights and node weights, this kind of query is illustrative of problems that involve uncertainties. However, in this implementation, the representation of the fuzzy variable weight is based on classical set theory where the membership can be clearly defined by a set. It simply performs a low level fuzzy query.

In the future we plan to continue research on the use of CLIPS in spatial data analysis. We intend to investigate also the use of FuzzyCLIPS for high level information queries, in which the representation of weights information is based on the concept of fuzzy set theory.

6. References

- [1] M. A. Cobb, "Modeling Spatial Relationships Within a Fuzzy Framework," *Journal of the American Society for Information Science*, Vol. 49, No. 3, 1998, pp 253-266.
- [2] CLIPS Reference Manual, Version 6.10, August 5th, 1998.
- [3] R. M. Wygant, "CLIPS - A Powerful Development and Delivery Expert System Tool", *Computers in Engineering*, Vol. 17, No. 1-4, 1989, pp 546-549.
- [4] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, Vol. 26, No. 11, 1983, pp 832-843.D
- [5] S. Winter, "Topological Relations between Discrete Regions," *Proceedings of 4th International Symposium, SSD'95*, Portland, ME, USA, August, 1995, pp 310-327.
- [6] J. Peuquet and Z. Ci-Xiang, "An algorithm to Determine the Directional Relationship Between Arbitrary-shaped Polygons in the Plane," *Pattern Recognition*, 20(1), 1987, pp 65-74.

Proceedings of the International Symposium on Distributed Objects and Applications

September 5 – 6, 1999
Edinburgh, Scotland

Supported by OMG (Object Management Group)

Edited by

Zahir Tari
Robert Meersman
Richard Soley
Omran Bukhres


IEEE
COMPUTER
SOCIETY



Los Alamitos, California

Washington • Brussels • Tokyo

Distributing Mapping Objects with the Geospatial Information Database

Miyi Chung, Ruth Wilson, Kevin Shaw
Naval Research Laboratory
Mapping, Charting & Geodesy
Stennis Space Center, MS 39529
{chung, wilson, shaw}@nrlssc.navy.mil

Maria A. Cobb
Dept. of Computer Science & Statistics
University of Southern Mississippi
Hattiesburg, MS 39406-5106
maria.cobb@usm.edu

Abstract

The Geospatial Information Database (GIDB) is an implementation of ongoing research in object-oriented geographic data modeling at the Naval Research Laboratory's Mapping, Charting & Geodesy Branch. The GIDB has evolved over the last five years from the initial memory-resident application involving vector mapping data, to the current state-of-the-art system of a distributed object-oriented database with web-based viewing capabilities for vector, raster, hypertext and multimedia data, as well as remote updating of vector data. The use of geographic data is becoming pervasive across many disciplines. At the same time, end users are becoming increasingly dependent upon the web as a source of readily available, easily accessible information. We believe these two factors necessitate the development of systems capable of the immediate distribution and access to complex spatial data objects. In this paper, we present the design strategies and implementation architecture of the GIDB.

1. Introduction and background

The Defense Modeling and Simulation Office (DMSO) and the National Imagery and Mapping Agency (NIMA) sponsored a FY 94 pilot project at the Naval Research Laboratory (NRL) at Stennis Space Center to produce a prototype object-oriented (OO) database with NIMA's first digital vector mapping prototype, Digital Nautical Chart (DNC). NRL teamed with the University of Florida to develop this prototype.

DNC was the first Vector Product Format (VPF) [5] dataset implemented by NIMA. VPF is a relational file format that represents geographic entities (features), along with their spatial and non-spatial attributes, through the use of tables. The project addressed the following areas: topological support among coverages, potential for not duplicating features among coverages, improved updating potential, and increased access speed. At project

completion, key findings reported to NIMA included the following:

- An OO Database with DNC information content could be implemented.
- Feature content was only stored once as compared to the repeated storage of some features in the conventional DNC.
- Direct updating of features and attributes was demonstrated.
- An order of magnitude speedup was demonstrated in feature access time.
- Features and attributes can easily be modified via a point-and-click interface.

NRL extended the prototype OO structure in FY 95 to accommodate multiple VPF databases as well as two different OO database management systems (ODBMS) [12]. This prototype is called the Object-Oriented Vector Product Format (OVPF), and represents a transformation of NIMA VPF relational databases into an OO structure. The OVPF allowed NIMA a rapid look at the potential benefits of OO approaches for allowing Department of Defense users to ask for information from NIMA that spans multiple databases [14].

During FY 96, much progress on conflation [2], [7], [8] and the base research for an integrated OO framework [13] to support multiple NIMA data types was accomplished. In FY 97, NRL developed the first prototype of the integrated OO framework, as well as developing a prototype OO Digital Nautical Chart updating system for NIMA. This system showed a 24:1 speedup over NIMA's current approach. Also, NRL developed the initial CORBA interface for the integrated framework to allow improved electronic dissemination of digital mapping objects [3], [16].

Marine Corps Warfighting Lab (MCWL) funded the NRL to develop the initial Geospatial Information Database (GIDB) during FY 98 and FY 99 to support an Integrated Marine Corps Multi-Agent Command and Control System (IMMACCS). This initial prototype demonstrated an ability to actively manage NIMA

mapping data in an object-oriented manner that could be interfaced with components developed by Stanford Research Institute, California Polytechnic Institute, and NASA's Jet Propulsion Laboratory. The GIDB demonstrated that this integrated database could be viewed and engaged in both 2D and 3D via a simple Internet browser. The GIDB was a component of the IMMACCS successfully used in the Urban Warrior Advanced Warfighting Experiment in March 99.

The underlying motivation for having an Internet-based Java client access our OO mapping database is to give end users the ability to access and use NIMA data quickly and efficiently. Currently, users of NIMA data must have resident on their own computer systems software to view the data, and must obtain the data on CD-ROM or other storage media. However, given NIMA's role as the primary geographic data distributor for the Department of Defense, it is clear that electronic dissemination and remote updating of NIMA's digital products is highly desirable. To this end, the GIDB allows any user with a Java-enabled web browser, such as Netscape 4.0, to access our database over the Internet and display NIMA map data available in their area of interest. Additionally, privileged users, known as data co-producers, are given the ability to perform remote updates on the data.

This paper presents the design and underlying architecture of the GIDB as used in the IMMACCS project described above. The remainder of this paper is organized as follows. Section 2 provides a high-level description of the overall architecture, followed by emphasis on the distributed aspects of the architecture in section 3. Section 4 focuses on the web applet, which is the mechanism by which remote viewing and updating of information is performed. Section 5 contains details of the network updating scheme, and includes a description of the IMMACCS updating test. In section 6, we present our concluding remarks and indicate directions for future work.

2. GIDB architecture

The Geospatial Information Database (GIDB) system has a client and server architecture. It is composed of server, interface and client modules. Currently, GIDB has three clients as shown in figure 1.

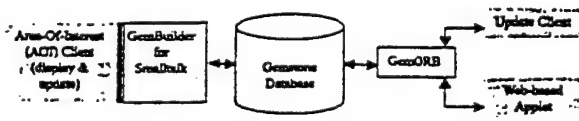


Figure 1. GIDB system component.

2.1. Server

Gemstone is a commercial-off-the-shelf object server that stores, manipulates, and processes objects referenced by each client. The server consists of two functional modules: storage of data, and manipulation or processing of data. Based on the request from each client, the Gemstone server searches and retrieves only those objects that meet the requested criteria. Data search for retrieval is performed mostly on the server for all three clients. Gemstone is an intelligent object server; it knows its objects by name. Therefore, Gemstone maintains its own object names. An object can be searched and retrieved by specifying its name. All disk-based systems involve a fetch at a page level. Sometimes the exact content of a page may not be explicitly known for most servers. However, for Gemstone as an object-based system, a content of a page can be known at an individual object level. A processing to determine what is on the page can take place on the server rather than on the client.

A server maintains its VPF data in the following manner. Entry points for all three clients are at the VPFDATABASE class level. VPFDATABASE class is the superset of all VPF data. VPFDATABASE class has a class variable or a global dictionary called Databases that contains all instances of the VPFDATABASE class. A root entry to any feature access begins with the Databases of VPFDATABASE class.

VPF data has a hierarchical structure. A database is used to group a set of data that is used for a specific purpose, e.g., Digital Nautical Chart (DNC) for navigation. It contains a collection of libraries. A library is used to group those features that are collected at a certain scale over a certain region. There may be some overlap or complete containment of one library to another. However, each library is unique based on the region and scale. Each library subsequently contains a collection of coverages, where each coverage contains those features that are related by a common theme, e.g., transportation or cultural.

A database, library and coverage triad, represented as VPFDATABASE, VPFLibrary, and VPFCoverage classes uniquely identifies a feature. A feature is defined at a coverage level. Due to tabular storage constraints, VPF data structure groups data at yet another layer, the tile. Each tile consists of some geographic extent in a minute by minute or a degree by degree manner. Figure 2 shows an example of a VMAAWE database having a collection of libraries such as Presidio, Oak Knoll, etc. A Monterey library consists of coverages or themes such as population, transportation, etc.

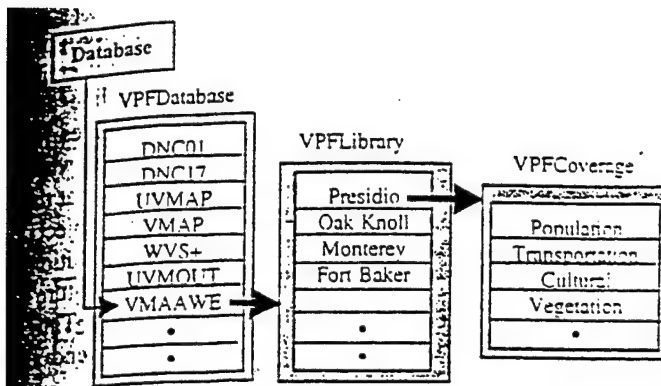


Figure 2. Structure of Gemstone database classes.

A server is designed to use a coverage as the minimal grouping level for features or objects. Every instance of a VPFCoverage has an instance of a dictionary collection called covQuad. A covQuad maintains all instances of VPFSpatialDataManager for a given coverage. A VPFSpatialDataManager class represents a spatial indexing scheme for organizing spatial data. The GIDB system uses a quadtree spatial indexing scheme to provide a hierarchical clustering of data based on the geographic area. A quadtree recursively divides an area into quadrants, each of which is called a quadcell. A detailed discussion of a quadtree indexing scheme can be found in [11]. In the GIDB system design, a class named VPFSpatialDataManager is created to represent a quadtree indexing scheme. All spatial objects or features are stored and indexed in a quadtree. An insertion of an object into a quadtree is based on the bounding box of the object. A quadcell that will minimally contain the boundingbox of an object is selected to store the object.

VPF data has three types of features: point, line and area (polygon). For efficient and faster access and retrieval, each feature type has a unique instance of a quadtree, i.e., there are three instances of VPFSpatialDataManager class. Therefore, a covQuad has three instances of VPFSpatialDataManager keyed by the feature type.

Any data access and retrieval begin by specifying the database, library and coverage. A feature retrieval may specify a part of an area or an area of interest (AOI) by specifying a geographic extent or the entire area of the database and library. This request is sent to the appropriate instance of VPFSpatialDataManager for actual feature retrieval.

2.2. Interface

Both GemBuilder for Smalltalk and GemORB are commercial-off-the-shelf products. Both components provide an interface to the Gemstone object server. GemORB is a Common Object Request Broker Architecture (CORBA) 2.0-compliant object request

broker. GemBuilder for Smalltalk is an interface between the AOI-based client and Gemstone.

GemBuilder for Smalltalk maintains its own object names as well. To establish a connection between an AOI-based client and Gemstone, a naming convention of each object must be resolved. In other words, the client and server must have an agreement on how to reference an object by name. GemBuilder for Smalltalk provides those classes that institute a convention for referencing the same objects between the AOI-based clients and Gemstone. For this reason, GemBuilder for Smalltalk requires some knowledge of the database design and implementation; the level of required detail is client dependent.

GemORB establishes a connection to the object server through CORBA-compliant communication. For more on CORBA, see [9], [10], and [15]. GemORB provides those classes that represent and implement CORBA. Unlike GemBuilder for Smalltalk, a connection via GemORB does not require an in-depth knowledge of the system design and implementation. An Interface Definition Language (IDL) file defines a correct mapping of objects between the client and the server. An IDL file also defines operations or methods that are available for clients to invoke on the server. Since GemORB is based on CORBA, all the benefits of interoperability among programming languages and platforms apply. Figure 3 shows the difference between the GemBuilder for Smalltalk and GemORB based applications.

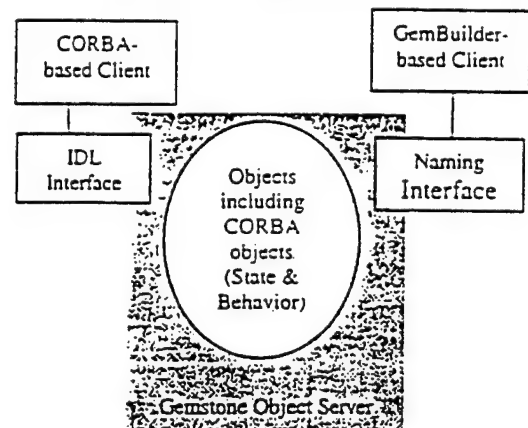


Figure 3. Gemstone object server configuration.

2.3. Client

An AOI client connects to the object server through GemBuilder for Smalltalk. This client mainly populates, maintains, updates and exports data. This client is tightly-coupled to the Gemstone design of data, i.e., class definition, class states and behaviors. A similar class definition is used between Gemstone and an AOI client;

an AOI client closely replicates the object server's design of data. Due to the data encapsulation property, a reference to an object implies a reference to a self-contained object. For those objects that are maintained and managed by GIDB, a self-contained object can consist of a large web of references to other objects, i.e., pointers. Since an object referenced by an AOI client is self-contained, AOI clients primarily request the object server to search and return objects. In most cases, AOI clients then process the data on the client side.

A GemORB based client, on the other hand, does not reflect server's design. These clients minimize information maintenance and storage by relying on the object server to be a centralized data storage as well as a centralized processing center. A GemORB client request for information expects the object server to search and completely process information. A client will receive fully processed information that can be readily used without further processing. These clients expect an answer to a question: AOI clients expect from the object server those parts that are needed to solve and derive the solution. Thus, AOI-based clients can be considered as "fat clients," because the implementation details are replicated on the clients, adding storage requirement. They are expected to process the information retrieved from the object server. The GemORB-based clients, however, are considered as "thin clients," because the implementation of those objects is not represented on the clients; there is not much processing involved on the client side. This paper will concentrate on the clients using GemORB.

3. Distributed architecture background

Information distribution of updates is a major concern among data users. This is especially true for NIMA users since NIMA is the only authorized data producer for military users. Changes must be captured, validated, and then incorporated into systems that utilize the data for any military operations. Furthermore, it is essential that military users have the latest data available, as well as a synchronized view of the space by various and multiple users.

There are three types of fundamental changes: geometrical, topological, and attribute. However, a geometrical change implies a topological change; a topological change cannot take place unless there is a geometrical change. Thus, this paper considers only two types of basic changes: geometrical and attribute.

There are two concerns in maintaining vector data when updates take place: (1) maintenance of internal topology and (2) distribution of updates to the users. NIMA produces Vector Product Format (VPF) data. These vector data have an internal topology called winged-edge topology [5]. This topology provides adjacency and contiguity information at a primitive level,

where one or more primitives define a feature. Five types of primitives are used in defining a VPF feature: entity nodes for point features, connected nodes for line and area features, edges for line and area features, faces for area features, and rings for area features. Therefore, when a geometrical change occurs, there is a possibility that the underlying topology may change for neighboring objects. This leads into the next relevant issue of distribution. When a feature changes, its neighboring feature's topological relationships may also change. Thus, determining what and how much information to send as an update is an issue.

Most applications that handle VPF data understand each feature in the context of its thematic layer, i.e., database, library, and coverage. The underlying relational structures that support such a layer consist of many tables and joins. An update can only be considered at a coverage level, although an update takes place at a feature level. A coverage may contain hundreds of features. However, due to the relational format of the vector data, an update implies potential changes to an entire coverage. Network utilization is non-optimal due to the redundant information, i.e., information that has not been changed will also be sent as a part of the update. With a shift from relational to object-oriented paradigm for VPF in GIDB, a map-entity or feature-level (object-level) update is implemented [1]. Object-level updating alleviates the two problems that were encountered in updating traditional VPF relational databases. Only an object that has been modified will be distributed. Any change in topology does not have to be transmitted as a part of an update. The GIDB server automatically triggers winged-edge topology checking whenever a new primitive or a change in a primitive is detected. Since only an object that has been updated is distributed, the minimal information of only the object that is immediately changed is distributed to the DoD services. Of course, this implies that the candidates for object-oriented feature-level updates require the receiving application to be object-oriented, as well.

In this scenario, one can imagine a server and client architecture in which NIMA is the centralized data source and all DoD services become clients as data users. Another scenario that can be considered is that of a distributed system. A distributed system is composed of systems that logically belong together, but that are in different physical locations. A distributed system is ideal when data sharing is required while some local control is maintained [6]. In the updating design, a client has the control to accept the update or reject the update from the server. Each client to the NIMA server has captured a "mini-world" or a geographic portion of the world. A combination of clients to the NIMA server could possibly become the distributed system, each providing other applications and systems with the latest data from NIMA covering certain geographic regions. In implementing this

distributed system. CORBA communication network has been used. This is shown in figure 4.

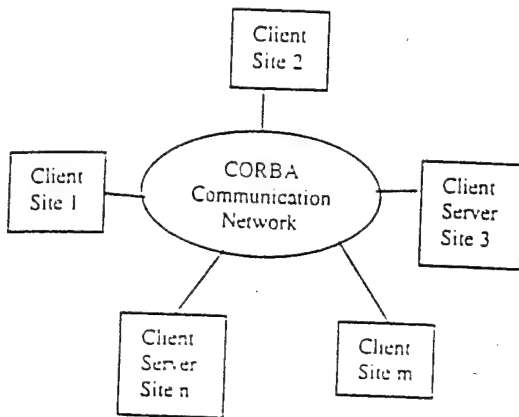


Figure 4. Distributed architecture example.

Based on figure 1, a web-based client would be a client only, whereas the update client could be both client and server. Both clients use CORBA as the communication network. CORBA enables simple query and data transmission over a network.

4. Web applet

The goal of the project was to have online access to geospatial data such as raster images and vector features over the Internet. These geospatial objects would be retrieved from a GemStone server. Communication between the server and a client is accomplished using CORBA-compliant vendor ORBs. The use of VisiBroker on the client side and GemORB for the database server is completely transparent to anyone accessing the applet. Figure 5 shows the basic architecture of our system. A web-based client has capabilities to display, select, and query objects interactively.

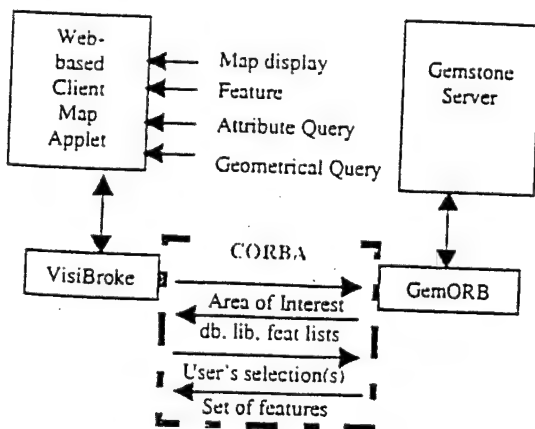


Figure 5. Basic system design.

A dialog between a server and a client begins when a client specifies an AOI. This specification is currently made by providing latitude, longitude and a radius. Currently, an interactive AOI selection directly from a map is disabled. This interface is shown in figure 6. Well-known and frequently accessed area names have been provided. The listed names are those AOIs that were used during the Urban Warrior experiment.

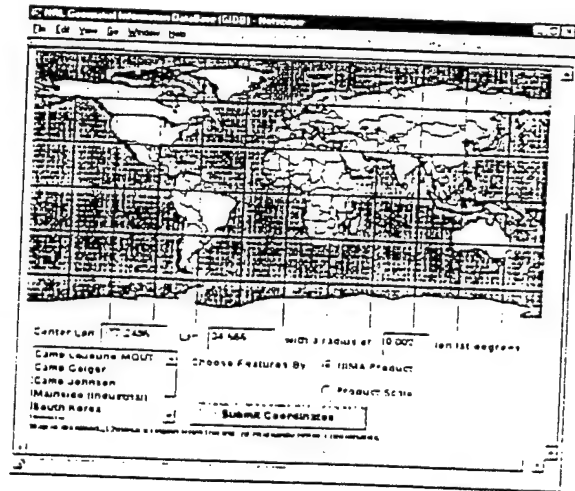


Figure 6. Area of interest screen.

Upon receiving an AOI request, the server retrieves all data sets that contain information over the AOI. A user will be able to tailor the display by selecting only those data sets that are of interest. The selection process is shown in Figure 7.

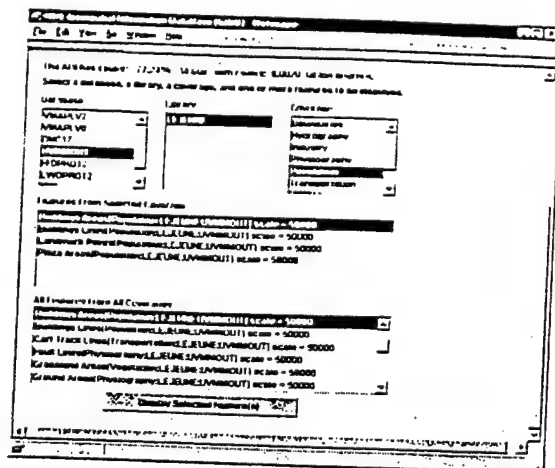


Figure 7. Database, library, and feature selection screen.

The selected features are sent to the server, and the server returns those objects that meet the selection criteria. These objects are displayed as shown in figure 8. Simple queries such as object attribute retrieval, as well

as advanced queries such as geometrical queries, e.g., objects within a distance of X kilometers, can be executed.

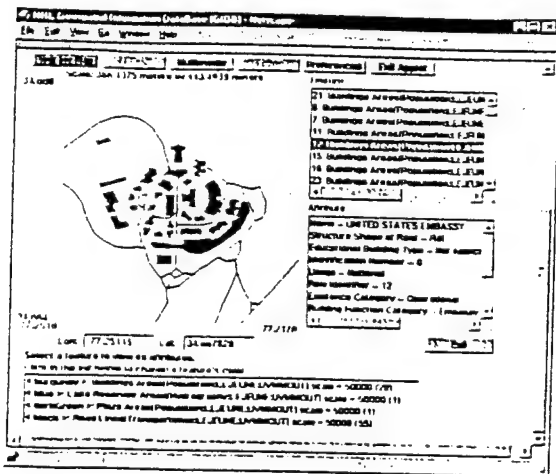


Figure 8. Simple query screen.

5. Information distribution

5.1. Basic system components and design

For information distribution from a GIDB server to a GIDB client, both the server and the client applications are identical. A peer-to-peer system configuration for CORBA has been implemented. A well-defined set of methods in an IDL file is used between systems to query and retrieve objects. Any system can become a server and client based on the needs.

The designation of server or client is based on the role a GIDB system assumes. A GIDB system can be a server to a suite of clients for a certain type of dataset. However, the same GIDB system can be a client to some other server for another dataset. This capability demonstrates a "smart client pull" information flow. The following assumptions are made in using this "smart client pull":

- A server is up and running constantly. Clients are on-line as needed.
- Both server and client maintain a log. A server maintains an update history log. The client maintains a client history log. These are represented in figure 9.
- A client initiates an update check. When a user logs onto the Gemstone server, a request is sent to the server via ORB-to-ORB communication to check for any update. A check on whether a client needs an update from a server's change log is based on a timestamp and the state of the feature in terms of its location and attributes.

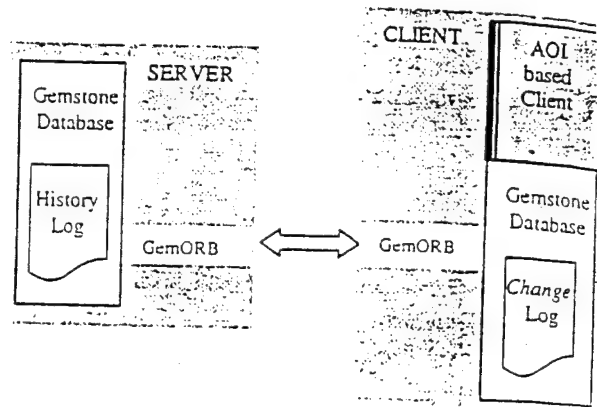


Figure 9. System components for updating.

This "smart client pull" allows background processing to automatically update the changes from the selected server. Based on this assumption, an interactive processing from the user is not required to initiate the update. It is also possible to have no user interaction for the actual update process: the system could be set up to automatically update the changes based on well-defined criteria. However, the current implementation allows a user to select which updates to perform, if any.

5.2. Update history log design

The GIDB application records all updates in a history log. The history log is maintained as a class variable to VPFDatabase and can be viewed by inspecting 'VPFDatabase historyLog.' The format of the history log is shown in figure 10.

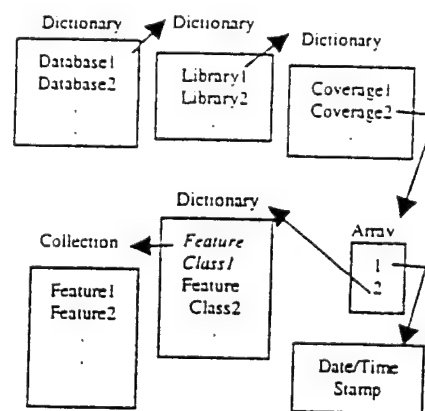


Figure 10. History log format.

When a feature is updated, an instance of a CORBA VectorFeature as defined in the IDL file is created and added to the appropriate feature collection in the history log. The coverage date/time stamp in the history log is changed to reflect the date/time that this feature was updated. Thus, the coverage date/time stamp reflects the

date/time of the most recent update that has occurred within the coverage.

5.3. Client history log design

When a client receives updates from a server, all updates are recorded in the history log as described above. In so doing, this client can then be a server to another client. In addition to recording the updates in the history log, a client also keeps a record of the updates in a client history log. The client history log is maintained as a class variable to VPFDATABASE and can be viewed by inspecting 'VPFDATABASE clientHistoryLog'. The format of the client history log is shown in figure 11.

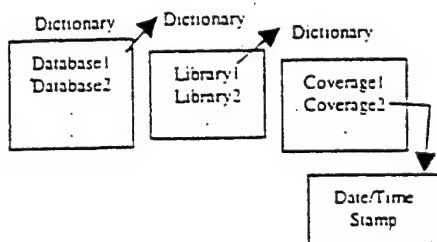


Figure 11. Client history log format.

The client history log records the date/time of the latest update for each coverage from the server. It is used to determine whether any updates have occurred since the last time the client was updated by the server.

5.4. Network update implementation

When a client logs on, the system automatically sends a CORBA request to the server for a list of available updates. During the login, the client invokes the server-side method `getUpdateLogFromServer`. This server-side method checks the server history log for updates. A list of strings comprised of database, library, and coverage names with timestamps, such as 'db1-lib1-cov1-01/27/99 13:37:37', is returned to the client. The client code then compares timestamps from the returned list of available updates with timestamps from the client history log to determine if the updates are needed on the client. If the client does need to be updated, a window appears allowing the user to select which updates to perform, as shown in figure 12.

The user may choose to update all, some, or none of the coverages. The items selected for update are then added to the client history log. As an item is being added to the client history log, the log is checked to determine if the coverage has been updated previously. If so, the timestamp for that coverage is updated, and the server timestamp is replaced with the previous update timestamp. If not, the server timestamp is replaced with

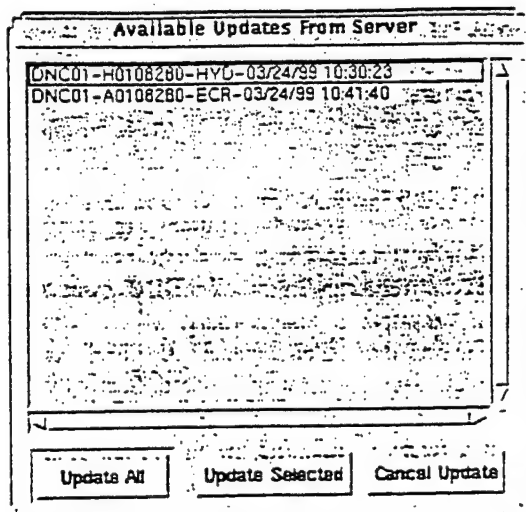


Figure 12. Available updates window.

the word 'none.' The timestamp replacement is used to prevent the server from sending back features that have already been updated. After the client history log is changed, the server-side method `getFeaturesToUpdate: updateSelections` is invoked. Figure 13 shows the exchange protocol between the server and client.

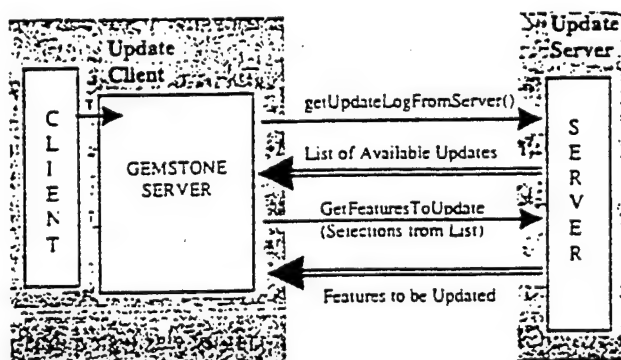


Figure 13. Smart client pull protocol.

For each item in the updateSelections list, the server finds the collection of updated features for the selected coverage. If the item in the updateSelections list has 'none' in place of its timestamp, then all of the features for this coverage are placed in the set of features to be updated. Otherwise, the timestamp from the updateSelections list coverage is compared to the timestamp of each feature. If the feature was updated at a later date and time than the coverage from the client, the feature is added to the set of features to be updated. This set of features to be updated is then returned to the client.

When the client receives the set of features to be updated, each feature in the set is updated. If the changeType is ADD, then a new feature is created based on the parameters of the VectorFeature. Otherwise, the

local client feature which matches the VectorFeature to be changed, deleted, or moved must be found in the client's database. The local client feature is found by using the VectorFeature featname and id. The oldAttributes and oldCoords are then compared with the local client feature to verify that the VectorFeature and the local client feature are indeed the same.

There are two potential sources for conflict in the search for a match. First, a client may have locally updated the feature. Since all GIDB systems have a capability to update feature data, a local update could have potentially taken place. A local update has precedence over the network update. Secondly, a feature can be uniquely identified by its database, library, coverage, feature class, and id. NIMA distributes its data with an additional identifier, an edition number. The latest edition will be a superset of all changes from the previous editions. The changes from one edition to another may coincide with the changes in the history log. However, the changes that take place by NIMA and the changes via GIDB are truly an independent effort. Because the edition numbers are not maintained by GIDB (assumed to have the latest released edition), there may be a mismatch in the edition of the server and client. Therefore, using the VectorFeature featname and id may not uniquely identify a feature. If the VectorFeature cannot be verified as a match to a local client feature, then the update for the VectorFeature will not occur.

When the feature has been validated, the local client feature is then changed, deleted, or moved based on the parameters of the VectorFeature. Recall that the update history log will be modified to reflect these updates from the server.

5.5. Near-real-time network update example

The GIDB network update capability was tested during the Marine Corps Warfighting Lab's Urban Warrior Advanced Warfighting Experiment in March 99. In preparation for the update test, a GIDB server was installed at NIMA in Bethesda, MD. A GIDB client was installed on the USS Coronado. On March 8, updates were made on the NIMA server for features in the DNC01 database, harbor library. In the Navigational coverage, several buoys were selected, and attribute values for one were changed. In the Hydrography coverage, attribute changes were made on bottom characteristic points. On March 10, the client on the USS Coronado received the updates from the NIMA server, and the updates were verified on the client. At the time of these experiments, the USS Coronado was at sea in the Pacific Ocean, and the CORBA communication occurred over a network, via satellite transmissions from ship to shore.

A similar test was performed on Thursday, March 11. The purpose of this test was to demonstrate that the client could be updated in near-real-time with updates from the NIMA server. Several features on the NIMA server were updated from a DNC01 general library and a VMAP Level 2 mission specific dataset. A few minutes later, the client on the USS Coronado was able to receive and perform these same updates. Quantitative measures of the update times were not taken during this initial testing of the GIDB network update capability. On average, retrieval of the list of updates from the server over a network took about 5 seconds. Retrieval and update of features took on average less than 30 seconds per feature, and this average decreased as the number of updated features increased. Performance of the network updating is extremely hard to measure due to the variety of parameters involved: the number of features being updated, the types of updates to be performed, the traffic on the network, the load of the client/server, etc. However, benchmarking studies will be performed in the near future.

6. Concluding remarks & future work

In this paper, we have shown how a web-based distributed system for retrieval and updating of mapping objects was implemented in the GIDB. The GIDB architecture relies heavily upon object technology and includes: a Smalltalk server application interfaced to a Gemstone ODBMS, Java/applet-based client applications, and CORBA middleware in the forms of VisiBroker and GemORB. The GIDB was the realization of our goal to have NIMA data available for electronic information distribution and updating, and played a significant role in the Marine Corps Warfighting Lab's Urban Warrior Advanced Warfighting Experiment earlier this year. The architectural components of the system worked well together; using Smalltalk as the server development environment allowed us to quickly prototype new capabilities, while Java provided the web-based capabilities for the user interface. CORBA proved an excellent choice to serve as a bridge between the two.

We are continuously working to expand the scope of capabilities of the GIDB, and to exploit the power of object-based technology for advancing the state-of-the-art in digital spatial data handling. Currently, we are working on completing the updating functionality and adding the use of map symbology for the display. For the near future, we plan to implement a rule-based distributed conflation system for the GIDB [4] that will automatically handle those cases in which a single feature is stored multiple times.

7. Acknowledgments

We would like to thank the Marine Corps Warfighting Laboratory (MCWL) and the National Imagery and

Mapping Agency (NIMA) for sponsoring this work as a part of the Urban Warrior Advanced Warfighting Experiment. We would like to thank our program managers at the MCWL, LtCol. Bott, Durham, and Stewart and Mr. Steve Hall and Mr. Gary Rogan at NIMA.

References

- [1] M.J. Chung, M.A. Cobb, K.B. Shaw, D. Arctur. An object-oriented approach for handling topology in VPF products. *Proc. GIS/LIS 95*, 1:163-174, 1995.
- [2] M.A. Cobb, M.J. Chung, V. Miller, H. Foley III, F.E. Petry, K.B. Shaw. A rule-based approach for the conflation of attributed vector data. *GeoInformatica*, 2(1):7-35, 1998.
- [3] M.A. Cobb, H. Foley III, R. Wilson, M.J. Chung and K.B. Shaw. An OO database migrates to the web. *IEEE Software*, 15(3): 22-30, May-June, 1998.
- [4] M.A. Cobb, F.E. Petry and K.B. Shaw. Uncertainty issues of conflation in a distributed environment. *Proc. GIS/LIS 98*, pp. 436-448 November, 1998.
- [5] Defense Mapping Agency (DMA). Military Standard: Vector Product Format. Draft Document No. MIL-STD-2407. Defense Mapping Agency, Fairfax, VA, 1993.
- [6] R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems*. Second ed., Addison-Wesley Publishing Company, 1994.
- [7] H. Foley, F.E. Petry, M.A. Cobb and K.B. Shaw. Utilization of an expert system for the analysis of semantic characteristics for improved conflation in geographic information systems. *Proc. 10th Intl. Conf. On Industrial and Engineering Applications of AI*, pp. 267-275, 1997.
- [8] H. Foley, F.E. Petry, M.A. Cobb and K.B. Shaw. Using semantic constraints for improved conflation in spatial databases. *Proc. 7th Intl. Fuzzy Systems Association World Congress*, pp. 193-197, 1997.
- [9] T.J. Mowbray and W.A. Ruh. *Inside CORBA: Distributed Object Standards and Applications*. Addison-Wesley, 1997.
- [10] Object Management Group. *The Common Object Request Broker: Architecture and Specification*. X/Open Company Ltd., U.K., 1996.
- [11] H. Samet. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Survey*, 16(2): 187-260, 1984.
- [12] K.B. Shaw, M.J. Chung and M.A. Cobb. Migration Process and Considerations for the Object-Oriented Vector Product Format to ObjectStore Database Management System. *Object Databases in Practice*, M.E.S. Loomis and A.B. Chaudhri, Prentice-Hall, 234-253, 1998.
- [13] K.B. Shaw, M.J. Chung and M.A. Cobb. Development of an integrated object-oriented framework for storage and query of multiple spatial data types of digital mapping information. *NRL Review*, pp. 112-115, 1997.
- [14] K.B. Shaw, M.A. Cobb, M.J. Chung and D. Arctur. Managing the navy's first object-oriented digital mapping project. *IEEE Computer*, 2(9):69-74, September 1996.
- [15] J. Siegel. *CORBA Fundamentals and Programming*. John Wiley & Sons, New York, 1996.
- [16] R. Wilson, M.J. Chung, T. Lovitt, B. Ray, M.A. Cobb and K.B. Shaw. Design of a java interface to a smalltalk oo mapping database utilizing CORBA. *Proc. Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 98)*, pp. 60-63, October, 1998.



URISA PROCEEDINGS

Papers from the annual conference

of the

URBAN AND REGIONAL
INFORMATION SYSTEMS ASSOCIATION

August 21 – August 25, 1999

Chicago, Illinois

Editor

Mark J. Salling, Ph.D.
The Urban Center
Levin College of Urban Affairs
Cleveland State University
Cleveland, Ohio

[Click Here for Abstracts](#)

[Click Here for Papers](#)

[Scroll Down for Table of Contents](#)

Kevin Shaw, Miyi J. Chung, Ruth Wilson, and Roy Ladner,
Naval Research Laboratory
Stennis Space Center, MS 39529
PH: (228) 688-4925 FAX: (228) 688-4853
{chung, wilson, ladner, shaw@nrlssc.navy.mil}

Maria A. Cobb
Department of Computer Science & Statistics
University of Southern Mississippi
Hattiesburg, MS 39406-5106
maria.cobb@usm.edu

J. Todd Lovitt
Planning Systems Inc.
115 Christian Lane
Slidell, LA 70458
todd.lovitt@psislidell.com

An Object-Oriented Database Approach for Urban Warfare

Abstract: The Naval Research Laboratory has developed a Geospatial Information Database (GIDB) using Object-Oriented Database, Common Object Request Broker Architecture (CORBA), and Virtual Reality Modeling Language (VRML) technology for the Marine Corps. The GIDB allows the integration of all digital mapping data types (e.g., raster, vector, text, audio and video) into a single database that can be signaled from a simple browser and directed by any area-of-interest. This technology will provide military and the commercial sector significant new abilities to select any urban area-of-interest and receive all related mapping objects via the Internet. The GIDB also enables constrained queries to allow Internet users to limit the provided objects to declutter the display (e.g., display only buildings within 500 meters of the coast for hurricane studies, along with related video and audio clips or only display roads where imagery is available as a background). The GIDB allows anyone with a simple Internet browser to interact with the database to signal portions of interest in the database to render itself in 3D, update itself, etc. The GIDB was utilized by the Marine Corps in the March 99 Urban Warrior exercise aimed at improving command and control in the urban environment.

INTRODUCTION AND BACKGROUND

The Defense Modeling and Simulation Office (DMSO) and the National Imagery and Mapping Agency (NIMA) sponsored a FY 94 pilot project at the Naval Research Laboratory (NRL) at Stennis Space Center to produce a prototype object-oriented (OO) database with NIMA's first digital vector mapping prototype, Digital Nautical Chart (DNC). NRL teamed with the University of Florida to develop this prototype.

DNC was the first Vector Product Format (VPF) (Defense Mapping Agency, 1993) dataset implemented by NIMA. VPF is a relational file format that represents geographic entities (features), along with their spatial and non-spatial attributes, through the use of tables. The project addressed the following areas: topological support among coverages, potential for not duplicating features among coverages, improved updating potential, and increased access speed. At project completion, key findings reported to NIMA included the following :

An OO Database with DNC information content could be implemented.

- Feature content was only stored once as compared to the repeated storage of some features in the conventional DNC.
- Direct updating of features and attributes was demonstrated.
- An order of magnitude speedup was demonstrated in feature access time.
- Features and attributes can easily be modified via a point-and-click interface.

NRL extended the prototype OO structure in FY 95 to accommodate multiple VPF databases as well as two different OO database management systems (ODBMS) (Shaw, et al., 1998). This prototype is called the Object-Oriented Vector Product Format (OVPF) and represents a transformation of NIMA VPF relational databases into an OO structure. The OVPF allowed NIMA a rapid look at the potential benefits of OO approaches for allowing Department of Defense users to ask for information from NIMA that spans multiple databases (Shaw et al., 1997).

During FY 96, much progress on conflation (Cobb, et al., 1998; Foley, et al., 1997) and the base research for an integrated OO framework (Shaw et al., 1997) to support multiple NIMA data types was accomplished. In FY 97, NRL developed the first prototype of the integrated OO framework, as well as developing a prototype OO Digital Nautical Chart updating system for NIMA. This system showed a 24:1 speedup over NIMA's current approach. Also, NRL developed the initial CORBA interface for the integrated framework to allow improved electronic dissemination of digital mapping objects (Cobb, et al., 1998; Wilson, et al., 1998)

Marine Corps Warfighting Lab (MCWL) funded the NRL to develop the initial GIDB during FY 98 and FY 99 to support an Integrated Marine Corps Multi-Agent Command and Control System (IMMACCS). This initial prototype demonstrated an ability to actively manage NIMA mapping data in an object-oriented manner that could be interfaced with components developed by Stanford Research Institute, California Polytechnic Institute, and NASA's Jet Propulsion Laboratory. The GIDB demonstrated that this integrated database could be viewed and engaged in both 2D and 3D via a simple Internet browser. The GIDB was a component of the IMMACCS successfully used in the Urban Warrior Advanced Warfighting Experiment in March '99.

The underlying motivation for having an Internet-based Java client access our OO mapping database is to give end users the ability to access and use NIMA data quickly and efficiently. Currently, users of NIMA data must have resident on their own computer systems software to view the data, and must obtain the data on CD-ROM or other storage media. However, given NIMA's role as the primary geographic data distributor for the Department of Defense, it is clear that electronic dissemination and remote updating of NIMA's digital products is highly desirable. To this end, the GIDB allows any user with a Java-enabled web browser, such as Netscape 4.0, to access our database over the Internet and display NIMA map data available in their area of interest. Additionally, privileged users, known as data co-producers, are given the ability to perform remote updates on the data.

This paper presents the design and underlying architecture of the IMMACCS as well as the GIDB architecture as used in the project described above. The remainder of this paper is organized as follows. The section immediately following provides a high-level description of the overall architecture, followed by emphasis on the distributed aspects of the architecture in the subsequent section. Following that, we focus on the web applet, which is the mechanism by which remote viewing and updating of information is performed. Details of the network updating scheme are then presented, including a description of the IMMACCS updating test. In the final section, we present our concluding remarks and indicate directions for future work.

IMMACCS ARCHITECTURE

In the post cold war era, there has been more focus on urban warfare. Urban warfare introduces some complication in that there are civilians, urban infrastructures and an open space with obstructed view. Marines are sent to urban settings to evacuate civilians under hostile situations. In preparation for potential urban warfare, MCWL has propelled an experiment and demonstration on how to conduct urban warfare using the latest technology available. In 1997, an exercise called Hunter Warrior took place with the focus on fighting smarter (more dependence on micro-chips), and using less physical force and (CNN, 1997).

Since many military operations take place via a chain of command, MCWL focused on the command and control activity within the Enhanced Combat Operations Center (ECOC). A specific requirement was imposed in supporting the experiment; the overall system was required to be object-oriented. MCWL believed that the object-oriented system would meet the overall objective in helping to effectively control the urban warfare. The IMMACCS consists of the following components: a real-time display, a 2-D viewer, communication backbone, intelligent agents, and a geospatial database. A real-time display was required to visualize the common tactical picture by officers in the ECOC as activities took place in the "battle space." Stanford Research Institute (SRI) developed and provided the 2-dimensional (2-D) viewer of the urban battlespace. Jet Propulsion Laboratory (JPL) provided a backbone (ShareNet) for all communication among the IMMACCS participants. Common Object Request Broker Architecture (CORBA) was the underlying medium that was used to exchange information among different components of the IMMACCS. California State Polytechnic Institute (CalPoly) developed and provided the intelligent agents, that were required to assist in the decision making process at the ECOC. SPAWAR's MCSIT ingested all track information from legacy command and control systems such as Joint Maritime Command Information System (JMCIS) and translated it into object-oriented format for the IMMACCS components to access and use. SRI's 2D-viewer (InCon™) as well as CalPoly's agents required the urban infrastructure to provide a visualization of the battle space as well as a capability to reason about the surrounding.

Both the intelligent agents and the display had two categories of information: static and dynamic. Dynamic information deals with tracking of movements such as troops, tanks, helicopters and a company of Marines. This information was provided to the IMMACCS through MCSIT as well as other GPS signals fed into the system provided by SRI. Static information provides both physical and built-up environments, man-made structures (e.g., buildings, facilities, and infrastructure) as well as natural structures (e.g., topography, vegetation, coastal lines). These data have geographic position as well as their occupancy in space. The term geospatial will be used to refer to the static information. The dynamic information is based upon the urban infrastructure in terms of its position, mobility and operation. Maps provide the static information, and are the basis for any strategic and tactical military operation, including urban warfare. NIMA has been designated as the VPF map provider to military services. NIMA provided the vector map information using relational tables. However, MCWL specifically required the overall system to be object-oriented. NRL ingested NIMA's vector data and provided the urban infrastructure information in object format. GIDB developed by NRL was used as the geospatial component of the IMMACCS.

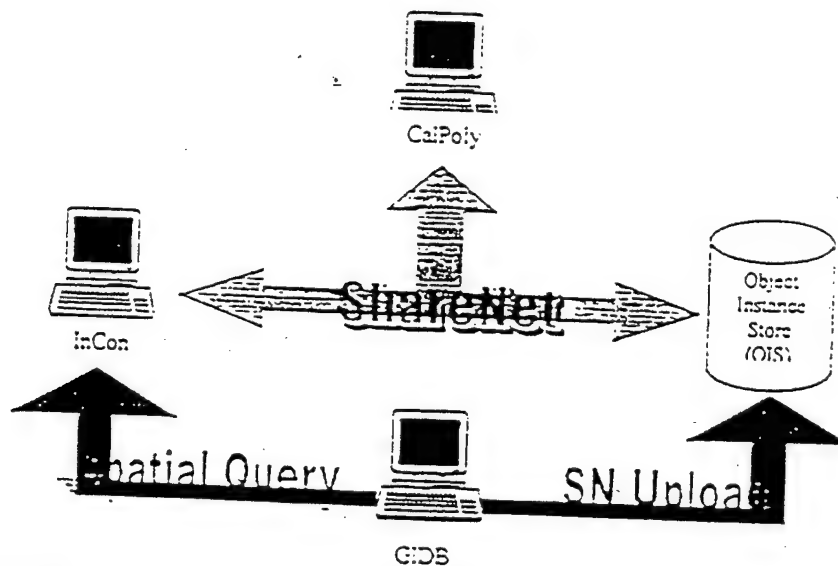
Both dynamic and urban infrastructure objects were persisted in the Object Instance Store (OIS) maintained by ShareNet. All objects must be in the OIS to be accessible by each system component. The OIS stores only the attributes of urban infrastructure objects; positional information of these objects are not maintained in the OIS. The InCon currently does not support vector maps for the infrastructure objects; an image is used as a reference map. Therefore, InCon needs to query the GIDB to determine which objects are available in the area of interest (AOI). Both the GIDB and the OIS maintain a global identification of each infrastructure object. When the GIDB provides the global identification of the objects to InCon, then InCon requests other information from OIS. This two-step query process is implemented because the attributes of the infrastructure objects as provided by NIMA are a subset of the attributes defined for each infrastructure object in the IMMACCS object model (IOM). The OIS provides more information relevant to the IMMACCS environment. Due to the imposed requirement of using object-oriented systems, CORBA was successfully utilized to create an integrated system, IMMACCS, from the different system components.

This section described the overall architecture of the IMMACCS system. Following is a list of GIDB capabilities that were provided to the IMMACCS as a part of the integrated system. Those capabilities that are inherent in GIDB are discussed in detailed under the GIDB architecture:

1. Transform the relational vector map information to object-oriented format.
2. Upload the urban infrastructure objects to the ShareNet's Object Instance Store via CORBA.
3. Allow InCon to perform spatial queries via CORBA.

Figure 1 shows the GIDB supportive role within the overall IMMACCS architecture.

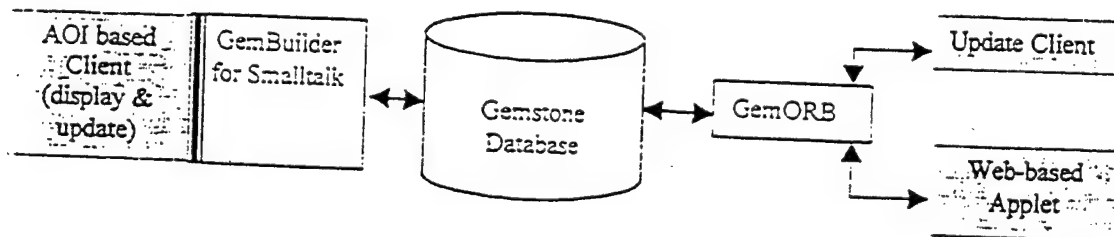
FIGURE 1
IMMACCS ARCHITECTURE



GIDB ARCHITECTURE

The Geospatial Information Database (GIDB) system is composed of several modules with three unique applications as shown in figure 2.

FIGURE 2
GIDB ARCHITECTURE



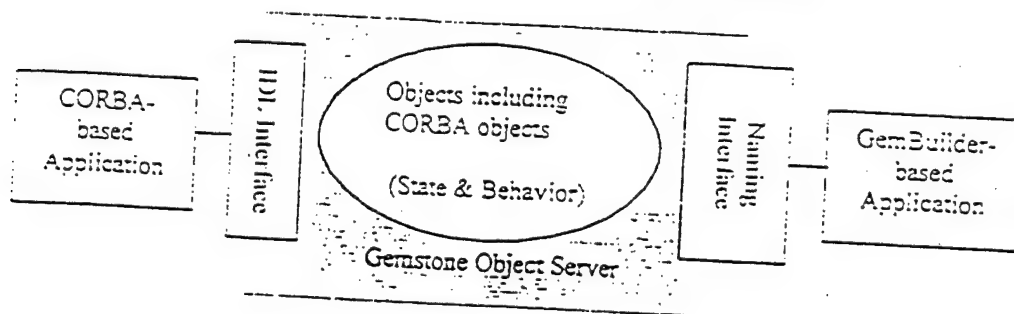
Gemstone is an object server that stores and manipulates objects as defined for each application. The server consists of two functional modules: storage of data, and manipulation of data. Based on the request from each application, the Gemstone server manipulates data to retrieve and return only those objects that meet the requested criteria. Data processing for retrieval is performed mostly on the server for all three applications. GemORB is a CORBA 2.0-compliant object request broker that works in conjunction with

Gemstone. For more on CORBA, see (Mowbray, 1997; OMG, 1996; Siegel, 1996). GemBuilder for Smalltalk provides the necessary interface between our Smalltalk application and Gemstone.

GemBuilder for Smalltalk provides all classes necessary for connection to the object server as well as object referencing for manipulation. The GemBuilder for Smalltalk allows connection to the server by a Gemstone defined mechanism with underlying system link and rpc invocations. An establishment of a connection requires naming resolution between the client and the object server. In other words, the client and server must have an agreement on how to reference an object by name.

GemORB establishes a connection to the object server through Common Object Request Broker Architecture (CORBA) compliant communication. GemORB provides all classes that represent and implement CORBA. The Interface Definition Language (IDL) file is created to allow for correct mappings of objects between the client application and the server application. The IDL file also defines the operations that are available to the client for invocation on the server. Since GemORB is CORBA-based, all the benefits of interoperability among programming languages and platforms apply. Figure 3 shows the difference between the GemBuilder for Smalltalk and GemORB based applications.

FIGURE 3
GEMSTONE OBJECT SERVER CONFIGURATION



The Area-of-Interest (AOI) based client is the application that uses GemBuilder for Smalltalk to connect to the object server. This application is used mainly for data population, data maintenance, data export, and data updates. An object may be represented as several nested objects of other object types. In fact, an object that reflects the actual implementation of the data is represented and used by AOI-based clients. The GemORB based applications do not reflect the implementation. These applications minimize information maintenance and storage by depending on the object server to maintain and store all pertinent information. These applications only request a minimal set of information to perform the requested function by a user. Thus, AOI-based applications can be considered as "fat clients," because the implementation details are represented on the clients, requiring heavy processing and more storage. The GemORB based clients are considered "thin clients," because these clients only request data that are required to perform its function; the implementation of those objects is not represented on the GemORB based clients. This paper will concentrate on the applications using GemORB.

An area-of-interest is implemented by using a spatial indexing scheme. The GIDB system uses a quadtree spatial indexing scheme to provide a hierarchical clustering of data based on geographic area. A quadtree recursively divides an area into quadrants, each of which is called a quadcell. A detailed discussion of a quadtree indexing scheme can be found in (Samet, 1984). In the GIDB system design, a class named VPFSpatialDataManager is created to represent a quadtree-indexing scheme. All spatial objects or features are stored and indexed in a quadtree. An insertion of an object into a quadtree is based on the geographic coordinates of the bounding box of the object. A quadcell that will minimally contain the boundingbox of an object is selected to store the object.

VPF data has three types of features: point, line and area (polygon). For efficient and faster access and retrieval, each feature type has a unique instance of a quadtree, i.e., there are three instances of VPFSpatialDataManager class. Any data access and retrieval begins by specifying the database, library and

coverage. A feature retrieval request may specify a part of the area by specifying a geographic extent or the entire area of the database and library. This request is sent to the appropriate instance of VPFSpatialDataManager for actual feature retrieval.

The GIDB provides a variety of capabilities due to the underlying object-oriented design. Aside from those capabilities that were stated specifically for the IMMACCS, the GIDB provides the following additional capabilities:

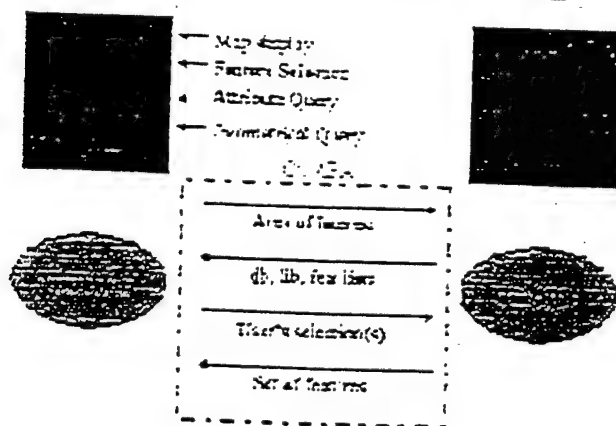
1. Stand-alone application as well as web-enabled application.
2. Spatial query (e.g., topological, geometrical and attribute-constrained queries) via Web browser and in stand-alone mode.
3. Network access with an optimized performance by the server responding only to the specified request from the user.
4. Request and visualize information by an area-of-interest.
5. Local dynamic updating or modification at a feature or an object level. Changes apply to both geometrical (e.g., add, delete, or modify location) as well as attributes changes.
6. Real-time updates from the data provider, e.g., NIMA.
7. Allow those changes and modifications to be exported back into VPF format.
8. Multi-media integration of audio, video, etc. based on the area of interest. For example, show a video clip or play an audio clip of some event in the area-of-interest.
9. Provide weather, news, etc., by exploiting Internet search engines by an area-of-interest.
10. Demonstrate 3-d rendering of an area-of-interest.

The remainder of this paper will focus on the web-enabled capability of the GIDB as well as the 3-D modeling effort.

INTERNET APPLLET

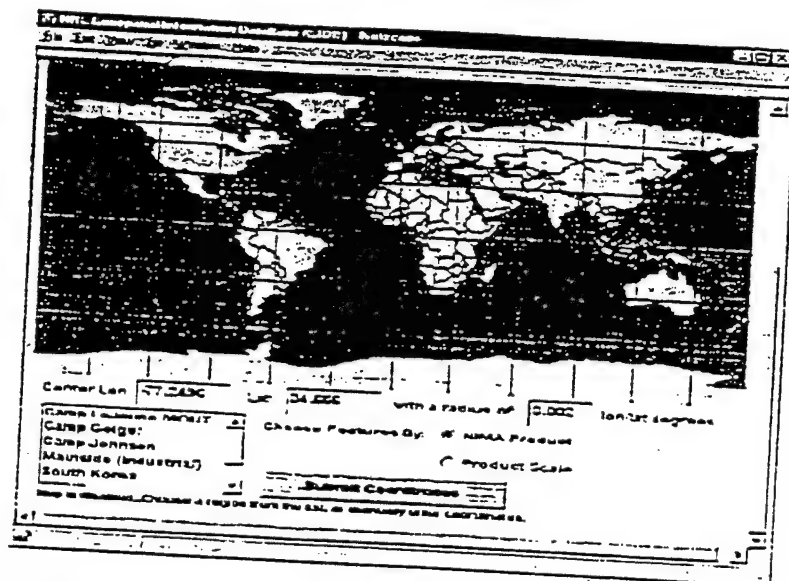
A Java-based Internet mapping client provides display and query capabilities for a set of geographic objects, such as raster images and vector features. These geographic objects are retrieved from a Smalltalk GemStone ODBMS acting as a server on a Unix machine. Communication between the Java applet, which is embedded in an HTML document on a web page, and the Smalltalk application is accomplished using CORBA-compliant vendor ORBs. The use of VisiBroker on the client side and Gem.Orb for the database server is completely transparent to anyone accessing the applet. Figure 4 shows the basic architecture of the system.

FIGURE 4
WEB-ENABLED SYSTEM ARCHITECTURE



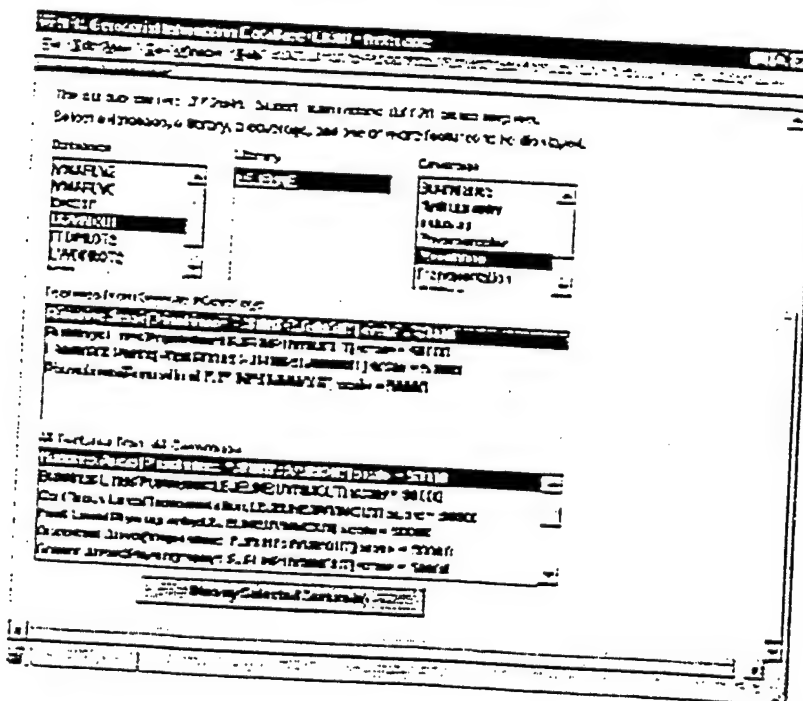
An area-of-interest is selected by entering a rectangular region specified by latitude, longitude and a radius. Alternatively, for some of the areas that were of interest to the Marines, a selectable list is provided. This can be seen in figure 5.

FIGURE 5
AREA-OF-INTEREST SCREEN



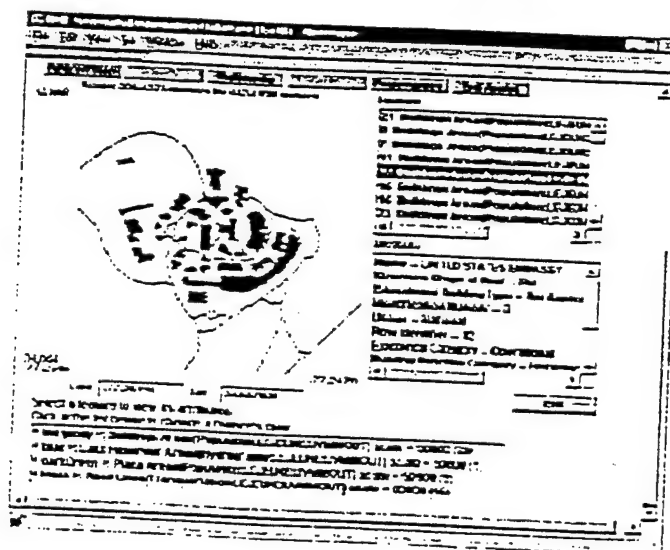
Once the area of interest has been selected, those VPF data that are available in the specified area are provided. Since Marines are familiar with the VPF products, they know which selections to make for their purpose. These lists are shown in figure 6.

FIGURE 6
VPF DATA SELECTION PROCESS



All objects that meet the initial selection criteria (e.g., 201, database-library-feature) are displayed. With the data that are returned, queries constrained by topology, geometry or attributes can be invoked, as shown in figure 7.

FIGURE 7
DISPLAY AND QUERY SCREEN

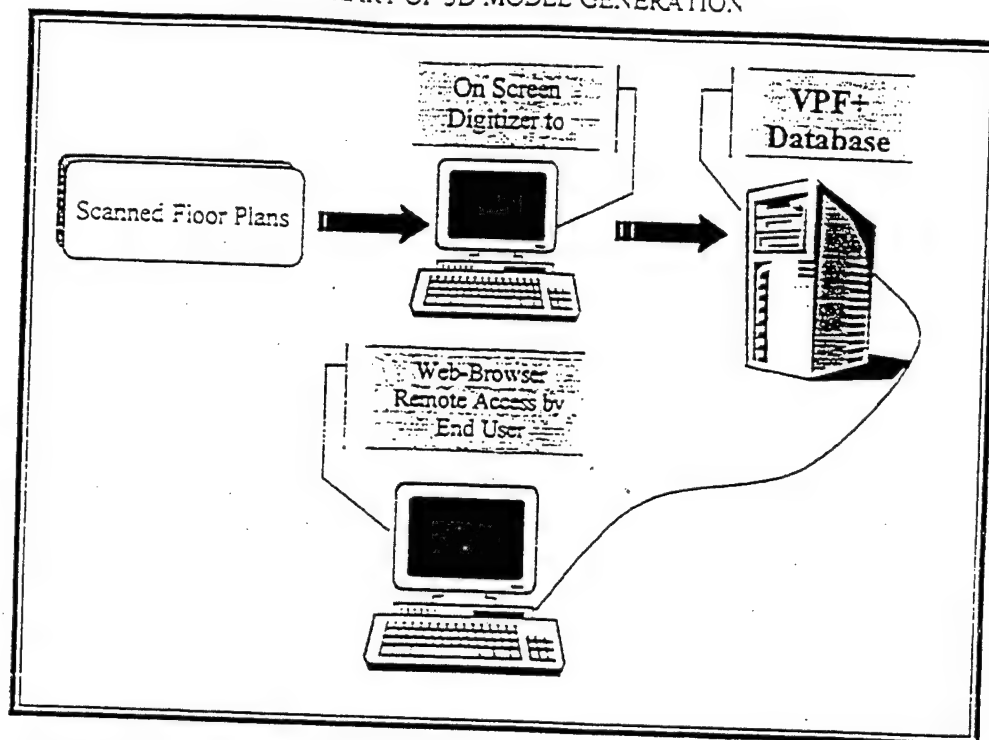


THREE DIMENSIONAL VISUALIZATION

A traditional, two-dimensional map display allows Marines to orient themselves in an area. However, in urban warfare, man-made features such as buildings become a threat of an enemy's hide-out. Also, Marines are required to clear a building as part of the process of conducting urban warfare. Therefore, a three dimensional view of a building, for example, provides valuable insight to those Marines as they take the necessary steps to enter and clear the building. Having recognized these needs by Marines, NRL provided the initial implementation of three-dimensional visualization using the VRML standard and commercial-off-the-shelf software. Once three-dimensional topology has been developed, spatial reasoning can be applied to assist in some operation analysis. To meet this objective, NRL developed VPF- (Ladner, 1998; Ladner, 1999). This work expanded NIMA's VPF format to include three-dimensional topology information.

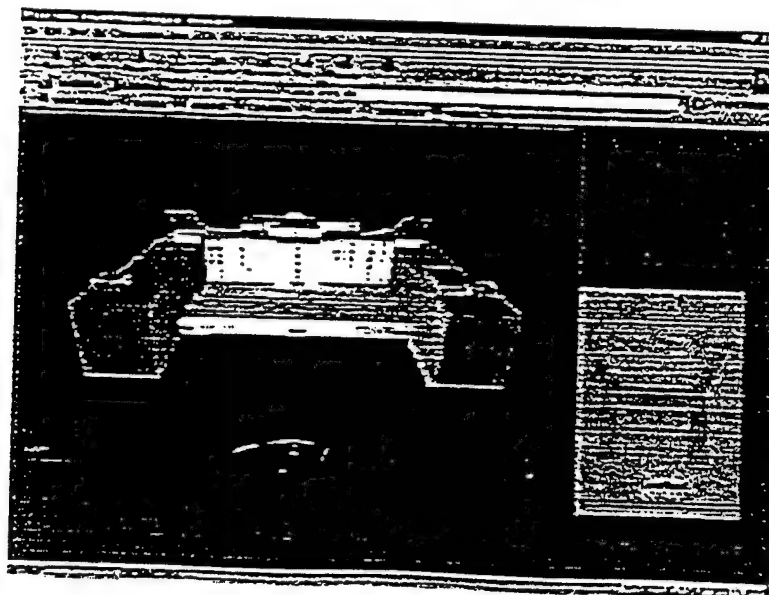
Figure 8 shows the flow chart of the steps taken to develop the VPF- database for the United States Public Service Health Hospital in Presidio, California, for the Urban Warrior project. Flat floor plans of the building were the only required inputs to this process. These plans provided detailed information about the building, such as floor layouts, dimensions, and means of entry. One of the VPF- tools we have developed is an on-screen digitizer that allows a user to interface with scanned floor plans to extract 3D geometric data and automate production. This allows accurate definition of the overall exterior of the building, and accurate placement of interior rooms, windows and doorways by defining the nodes, edges and faces that form the three-dimensional structure of the building. Using this tool and scanned floor plans of the hospital, 3D data for the building were generated and stored in the VPF- database.

FIGURE 8
FLOWCHART OF 3D MODEL GENERATION



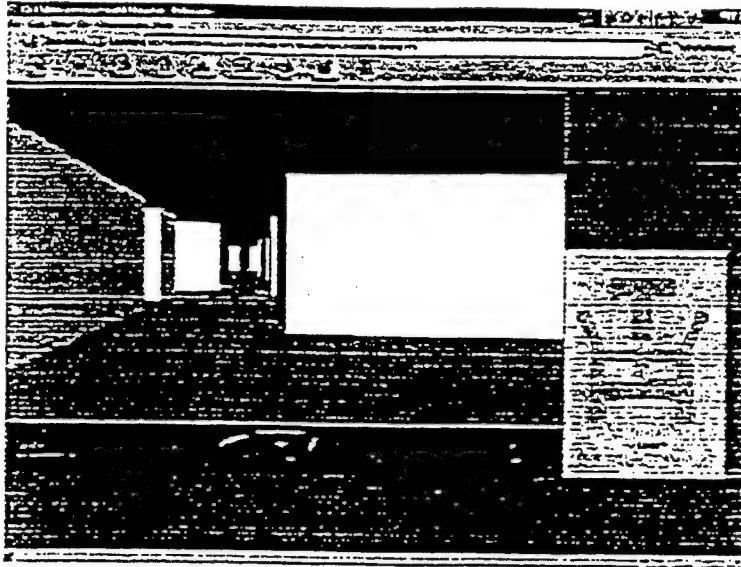
User interaction is through a web-browser equipped with a 3D graphic plug-in, and an application applet. Available interactions with the 3D virtual world include the ability to walk into buildings through doorways or climb through open windows, to look through open doorways and windows either from outside or inside buildings, and to enter rooms through interior doorways and climb stairs to different floors. A 2D map of the hospital is displayed adjacent to the 3D counterpart as shown in figure 9.

FIGURE 9
3D MODEL OF USPS PUBLIC HOSPITAL



Both a visualization of the 3D model at the current level of the Marine, and a 2D map that provides the current directional information are provided to enable those Marines inside a building to orient themselves within the overall context of the building. This can be seen in Figure 10. A pointer on the 2D map shows the user's location within the building and direction the user is heading.

FIGURE 10
TRACKING OF A MARINE INSIDE A BUILDING



A summary of 3-D display capabilities are provided in figure 11.

FIGURE 11
SUMMARY OF 3D VISUALIZATION

Three Views of the U.S. Public Health Service Hospital located at the Presidio, San Francisco, California. Figure 1 shows a typical 2D representation. Figure 2(a) shows a 3D-object model. Figure 2(b) shows a cut-away of the first floor including inside rooms, etc.

Figure 1.

Figure 2(a).

Figure 2(b).

CONCLUDING REMARKS AND FUTURE WORK

In this paper, we have shown how an object-oriented system can support military operation in an urban warfare environment. More specifically, the GIDB which focused on the geospatial information retrieval and visualization was presented in detail. Object-oriented technology was used in developing the application. Furthermore, the object-oriented nature of the IMMACCS project, along with the use of CORBA, enhanced interoperability among the separate system components. Functionalities such as web-enabling were also achieved. Three-dimensional support enhances the Marines' ability to operate more effectively with better environmental and situational awareness. Future work includes implementing VPF+ in the GIDB. Currently, the 3D generation is a stand-alone process; however, additional research and development for integrating the GIDB with the 3D rendering is in progress.

REFERENCES

- Chung, Miyi, Maria Cobb, Kevin Shaw, David Arctur, 1995. "An Object-oriented Approach for handling Topology in VPF Products," in GIS/LIS Proceedings. Vol. 1. Nashville, TN. pp. 163-174.
- CNN, 1997. "Military sees high tech future". <http://camaro.acq.osd.mil/at/mout.htm>
- Cobb, Maria A., Miyi J. Chung, Vincent Miller, Harold Foley III, Frederick E. Petry, Kevin B. Shaw, 1998. "A Rule-Based Approach For The Conflation of Attributed Vector Data," *GeoInformatica*, 2(1). pp. 7-35.
- Cobb, M., H. Foley III, R. Wilson, M. Chung and K. Shaw, 1998. "An OO Database Migrates to the Web," *IEEE Software*, 15(3). pp. 22-30.
- Cobb, M., F. Petry and K. Shaw, 1998. "Uncertainty Issues of Conflation in a Distributed Environment," *Proc. GIS/LIS '98*, Fort Worth, TX. pp. 436-448.
- Defense Mapping Agency (DMA), 1993. Military Standard: Vector Product Format, Draft Document No. MIL-STD-2407, Fairfax, VA: Defense Mapping Agency.
- IMMACCS. <http://netwarrior.jpl.nasa.gov>.
- Foley, H., F. Petry, M. Cobb and K. Shaw, 1997. "Using Semantic Constraints for Improved Conflation in Spatial Databases", *Proc. 7th Intl. Fuzzy Systems Association World Congress*, Prague. pp. 193-197.
- Ladner, Roy, Abdelguerfi, Mahdi, Shaw, Kevin, 1998. VPF+: A VPF Extension Providing 3D Modeling and 3D Topology. *Image '98 Conference*, Scottsdale, Arizona. pp. KB1-KB8.
- Ladner, Roy, Abdelguerfi, Mahdi, Shaw, Kevin. 1999. 3D Synthetic Environment Representation Using the "Non-Manifold 3D Winged-Edge" Data Structure. *Lecture Notes in Computer Science, Interoperating Geographic Information Systems, Second International Conference, Interop'99*, Zurich, Switzerland, pp. 305-314.
- Mowbray, Thomas J. and William A. Ruh, 1997. *Inside CORBA: Distributed Object Standards and Applications*, Addison-Wesley.
- Object Management Group. 1996. *The Common Object Request Broker: Architecture and Specification*, X/Open Company Ltd., U.K..
- Samet, H. 1984, "The Quadtree and related Hierarchical Data Structures," *ACM Computing Survey*. Vol. 16, No. 2. pp. 187-260.

Shaw, K., M. Chung and M. Cobb, 1998. "Migration Process and Considerations for the Object-Oriented Vector Product Format to ObjectStore Database Management System", in: Object Databases in Practice. Edited by M.E.S. Loomis and A.B. Chaudhri. Prentice-Hall, pp. 234-253.

Shaw, K., M. Chung and M. Cobb, 1997. "Development of an Integrated Object-Oriented Framework for Storage and Query of Multiple Spatial Data Types of Digital Mapping Information," NRL Review, NRL. pp. 112-115.

Shaw, K., M. Cobb, M. Chung and D. Arctur. 1996. "Managing the Navy's First Object-Oriented Digital Mapping Project," IEEE Computer. Vol. 2, No. 9. pp.69-74.

Wilson, R., M. Chung, T. Lovitt, B. Ray, M. Cobb and K. Shaw, 1998. "Design of a Java Interface to a Smalltalk OO Mapping Database Utilizing CORBA," Proc. Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS '98). Las Vegas, NV. pp.60-63.

ACKNOWLEDGEMENTS

We would like to acknowledge Marine Corps Warfighting Laboratory (MCWL), Program Element number 0603640M, for funding this project, with special thanks to Lt. Col. Bott and Lt. Col. Durham as program managers of the IMMACCS project.

A SPATIAL INDEXING FRAMEWORK USING A QUADTREE ORGANIZATION FOR GEOGRAPHIC DATA STORAGE AND RETRIEVAL

RUTH A. WILSON*, MARIA COBB†, MIYI CHUNG* AND KEVIN SHAW*

Abstract. The Digital Mapping, Charting and Geodesy Analysis Program of the Naval Research Laboratory has developed an object-oriented digital mapping database prototype, called the Geospatial Information Database (GIDB). This database application is capable of importing any of the National Imagery and Mapping Agency's (NIMA's) Vector Product Format data and converting the data into an object format. Other supported NIMA data types include Raster Product Format, Text Product Standard, and Digital Terrain Elevation Data. The GIDB supports multimedia data as well, including audio, video, and imagery (GIF and JPEG).

Our approach involves partitioning the globe into latitude-longitude cells, since retrieval of objects in the application is always based on a user-defined area of interest. Because any spatially referenced data can be indexed by the quadtree, spatial range queries, i.e., which objects are located within a particular area, are efficiently processed for the multiple data types stored in the GIDB. Each object is defined to have a minimum bounding rectangle, or latitude-longitude bounding box, which is used to determine its placement within a quadtree. In this paper, a brief description of the vector, raster, text, and gridded data types that are stored in the GIDB is presented, followed by a detailed description of the basic quadtree design. The utilization of the resulting quadtree organization is then outlined and discussed; specifically, the method by which objects are placed in the quadtree, as well as the algorithms for object retrieval, are analyzed.

1. Introduction. The Digital Mapping, Charting and Geodesy Analysis Program (DMAP) of the Naval Research Laboratory began an effort in 1994 to develop an object-oriented (OO) digital mapping database, called the Geospatial Information Database (GIDB) [1]. This database application is capable of importing multiple data formats and storing the data in a quadtree data structure for retrieval. The first data format to be implemented in the GIDB was Vector Product Format (VPF) data from the National Imagery and Mapping Agency (NIMA) [2]. The data is converted into an object format and persistently stored in a GemStone object-oriented database management system (ODBMS). Other supported NIMA data types include Raster Product Format (RPF) and Text Product Standard (TPS). The GIDB supports multimedia data as well, including audio, video, and imagery (GIF and JPEG).

These supported data types are stored in quadtree data structures within the GIDB. A quadtree is an indexing structure used for the storage and retrieval of two-dimensional data. The principle upon which it is based is simply the regular recursive subdivision of blocks of spatial data into four equal-sized cells, or quadrants. Cells are successively subdivided until some criterion is met, usually either: (1) each cell contains homogeneous data, e.g., a single "feature" for vector data, or rasters containing the same value (known as a region quadtree), or (2) a preset number of decomposition iterations has been performed [3]. Thus, the cells of a quadtree are of a standard size (in powers of two) and are non-overlapping in terms of areal representation. A tree structure is then constructed by arranging each cell as the parent of its component quadrant cells. This structuring leads to a tree whose nodes at any level of the tree are all of the same size, that size being exactly one-fourth the size of the nodes at the next higher level of the tree.

In this paper, a brief description of the various data types that are stored in the GIDB is presented, followed by a detailed description of the basic quadtree design. The utilization of the resulting quadtree organization is then outlined and discussed. The methods by which objects are placed in the quadtree, as well as the algorithms for object retrieval, are analyzed.

2. GIDB Data Formats. VPF features are geographic data objects composed of both spatial and non-spatial information. The non-spatial information includes source data, such as where the data were obtained, and attribute data. For example, if the VPF feature were a building, it would include, among other data, information about the building's height, width, name, and use. Spatially, each VPF feature contains latitude-longitude coordinate information. A VPF feature can be a point, a line, or an area feature. A point

* Naval Research Laboratory, Code 7440.2, Stennis Space Center, MS 39529. shaw@nrlssc.navy.mil.

† Department of Computer Science and Statistics, University of Southern Mississippi, Hattiesburg, MS 39406. Maria.Cobb@usm.edu.

consists of a single latitude-longitude coordinate pair. A line is a sequence of latitude-longitude points, and an area is a closed polygon region. Each line and area feature is defined to have a latitude-longitude bounding box, or minimum rectangular region that completely encloses the feature. For point features, the bounding box is determined using a small offset value.

Each VPF feature also has topology associated with it. Topology is the study of the characteristics of geometrical objects that are independent of the underlying coordinate system, such as adjacency and contiguity [4]. The topological information is provided to improve spatial analysis capabilities [5]. There is a difficulty in representing complex geographic data, such as VPF features, by decomposing the data to fit the flat-file structure imposed by a relational database model. A change in the topology of a VPF feature, for example, will result in changes to numerous tables in the relational model. With our object-oriented model in the GIDB, topological and other relationships among features can be handled more simply and directly due to encapsulation, inheritance, and polymorphism.

A minimum bounding rectangle (MBR) can be used as an approximation to an actual spatial entity [6]. An MBR is a minimum x-y parallel rectangular region necessary to completely enclose the spatial representation of data, as shown in Figure 1. The use of MBRs in geographic databases is widely practiced as an efficient way of locating and accessing objects in space [7]. A major advantage of an MBR representation is that all objects are represented as 2-D objects across which operations can be applied uniformly. Fortunately, VPF line and area data store MBR information in the form of the latitude-longitude bounding box. The spatial component of VPF point data is the coordinate itself. However, a minimal offset could be imposed with a radial distance using the coordinate location as a center, creating an MBR for the point feature. Hence, an MBR representation can be used for each VPF feature.

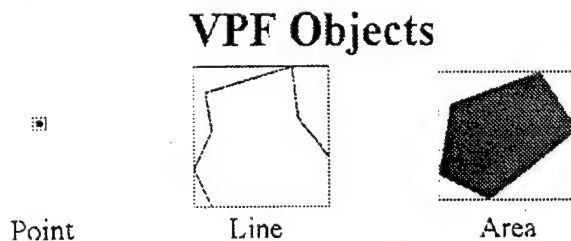


FIG. 1. Minimum Bounding Rectangles (MBRs) for VPF Features.

Although the GIDB was originally developed for storage and access of only VPF features, development has expanded to include other data types. NIMA's RPF data type is a format to store satellite imagery and digitized aeronautical charts. RPF data is composed of rectangular frames, and each frame is composed of smaller subframes. The subframe itself is a rectangular region, so it can be used as an MBR to correlate to the MBR of VPF features. Figure 2 shows a sample RPF image.

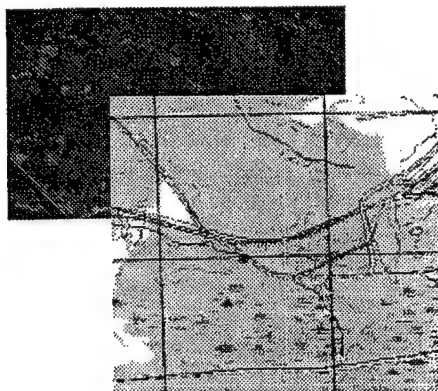


FIG. 2. Sample RPF Images.

NIMA's TPS data is a text file that contains relevant information for the user such as notices of changes or lessons learned from military missions. TPS data is in SGML format, and embedded within each data set is a simple browser to display the TPS data. Each TPS file has a gazeteer that includes an associated latitude-longitude position, so an MBR can be defined for it in the same way as for a VPF point feature.

An MBR is defined for multimedia data as well, representing the geographic region relevant to the audio clip, video clip, or imagery. Audio and video clips can be of any standard format. The imagery can be photographs, floor plans, or other items stored in GIF or JPEG format.

The MBR is defined for every object type to be stored in the GIDB. Consequently, the MBR can be used to determine how and where an object is stored in the database. Use of the MBR is the basis for the quadtree design and organization that is implemented in the GIDB.

3. Quadtree Organization. A quadtree is formed by the regular recursive subdivision of blocks of spatial data into four equal-sized cells, or quadrants. In our implementation, a cell is not created unless the cell has data or at least one of its subsequent children cells has data. Since cells are created only when they are needed, search time for objects is reduced and memory is not wasted. To determine whether an object is to be placed in a cell, its MBR is used. A simple intersection computation is calculated to determine a cell to house a given spatial object. Spatial data placement is determined based on the virtue of the MBR intersecting at least one of the dividing lines used to form children cells. An object is placed in the smallest cell that completely contains its MBR. In other words, an object is stored at the lowest level in which it fits in the quadtree. Figure 3 shows a sample subdivision of an image into quadrants and the resultant quadtree representation.

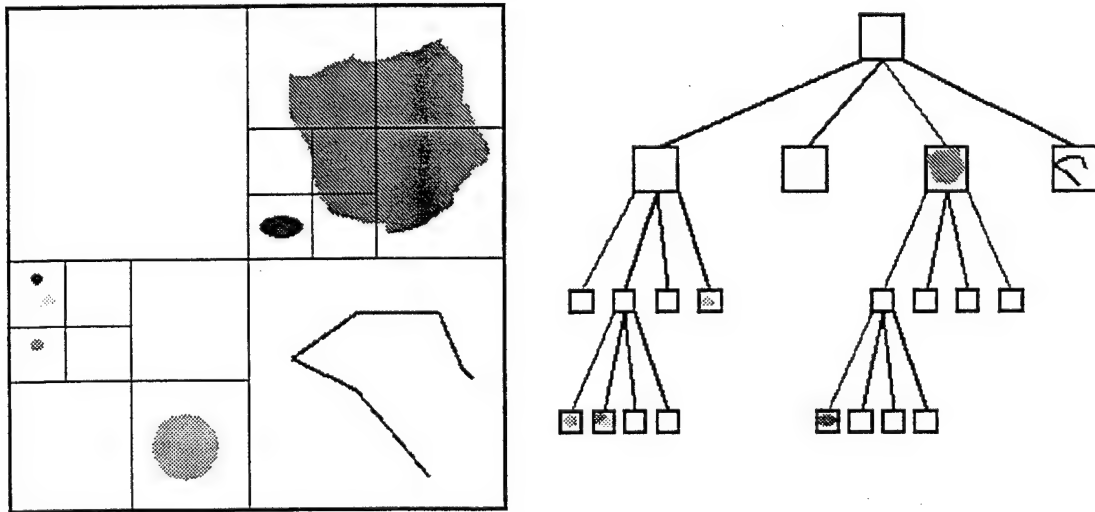


FIG. 3. Sample Subdivision into Quadrants and the Resultant Quadtree Representation

All of the data to be stored in our database is digital map data or data that can be geographically represented. Consequently, we define the root cell of our quadtree to be the minimum bounding square of the world. The root cell has a latitude-longitude bounding box of origin -180, -180 and corner 180, 180. Note that each cell of our quadtree will be a square region, which is a design consistent with most implementations of a quadtree structure. Within this application, the depth of a quadtree has been set to 20. The setting of a maximum depth implies that an extent of a cell at level 20 is about 1.4555 km x 1.4555 km. It is necessary to place an upper bound on the levels of the quadtree since a large number of VPF data are point features. An unbounded quadtree would be very deep to include cells that contain point features. Such a quadtree would also contain a large number of small children cells. Also, since a quadtree search always begins from the root of a tree, a limitation on the depth of a tree improves data search and retrieval

as well. The maximum depth of the quadtree can be easily increased in the future so cells at the lowest level cover a smaller geographic area.

A quadtree is implemented in our system using an object-oriented methodology. According to Frank and Egenhofer, there is a paradigm shift of spatial data from a relational data structure to an object-oriented representation [8]. Frank and Egenhofer concluded, "spatial data are too complex to be managed within a relational data model." For this reason, an object-oriented approach using the Smalltalk programming language is implemented in the GIDB application. An introduction to object-oriented class diagrams and terms developed and designed for the GIDB is presented in [9].

A quadtree is implemented in the GIDB application as *SpatialDataManager* and *SpatialDataCell* classes. As the name suggests, the *SpatialDataManager* manages the spatial tree indexing framework, or quadtree. The *SpatialDataManager* class has an instance variable *topCell* that represents the root of the tree. This instance variable is a pointer to the cell that represents the spatial bounds of all available data. The *SpatialDataManager* class has another instance variable *maxLevel* that gives the maximum depth of the quadtree. This maximum depth can be changed as more features are added to the quadtree so cells at the lowest level contain fewer features and cover a smaller geographic region.

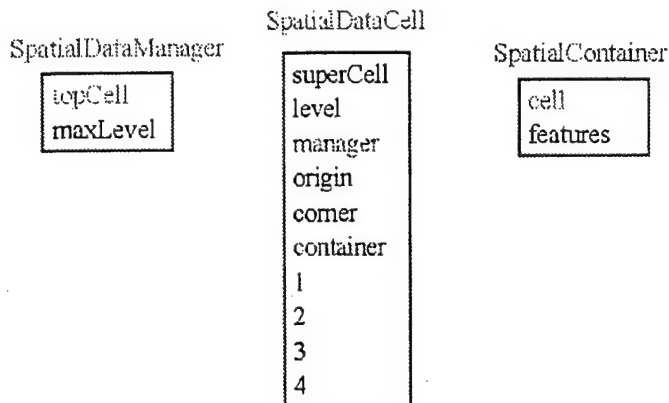


FIG. 4. *SpatialDataManager*, *SpatialDataCell*, and *SpatialContainer* Class Diagrams

A *topCell* is an instance of the *SpatialDataCell* class which has instance variables of *superCell*, *level*, *manager*, *origin*, *corner*, *container*, and children cells 1, 2, 3, and 4. A *superCell* is a pointer to the cell's parent cell. The *level* gives the level of the cell in the quadtree. *Manager* is a pointer to the quadtree itself. The *origin* and *corner* provide the latitude-longitude MBR of the cell. The *container* is a pointer to an instance of the *SpatialContainer* class that contains the features located in the cell. A cell can have a maximum of four children cells. When a cell is instantiated, the children cells 1, 2, 3, and 4 are initialized with a *nil* or null value. Each child cell becomes an instance of *SpatialDataCell* only when data needs to be added to the child cell or to one of its children.

A *container* is an instance of the *SpatialContainer* class that has instance variables of *cell* and *features*. The *cell* is a pointer to the cell that referenced the *container*. *Features* is a collection of all spatial objects for which the referencing cell is the lowest level cell completely containing each object's MBR. A diagram of the *SpatialDataManager*, *SpatialDataCell*, and *SpatialContainer* classes is given in figure 4.

4. Adding a Feature to the Quadtree. Data insertion into a quadtree begins with a search from the root of a tree. The following request is made: (*quadtree containerFor: aFeature boundingBox*) *addFeature: aFeature*. A *quadtree* is an instance of *SpatialDataManager*, so it represents the quadtree in which the item is being placed. It is asked to find a minimum cell that can fully contain the MBR of a spatial object, *aFeature boundingBox*. The *containerFor:* method first checks a cell to see if it can contain the feature's MBR by calling the method *aCell canContainBoundingBox: aFeature boundingBox*. If *aCell* can contain the feature's MBR, then the method *aCell containerForBoundingBox: aFeature boundingBox* is called. This method checks the cell and its children cells to determine the lowest level cell in which to

place the feature. Now that the appropriate cell is found *aFeature* is added to the *features* collection of the corresponding cell's *container*.

As mentioned previously, the MBR is used to add, retrieve, and remove any of the data type features from a quadtree. The calculation method for determining whether a feature's MBR intersects a quadcell's MBR is as follows:

featMBR intersects: cellMBR
 $\wedge ((featMBR\ origin \leq cellMBR\ corner) \text{ and: } (cellMBR\ origin \leq featMBR\ corner))$

The origin represents the lower left point of the MBR and the corner represents the upper right point of the MBR. Notice that this method is a simple, fast calculation to determine whether a feature should be within a quadcell.

5. Retrieving a Feature from the Quadtree. Data retrieval from the quadtree also begins with a search from the root of the tree. The method call *quadtree returnAllFeatures* will return a collection of all of the data objects that are located within the quadtree. The method call *quadtree returnNumberOfFeatures* will return only the number of data objects in the quadtree. To obtain a collection of all features that are located within a given area of interest (AOI), the method *quadtree returnSetOfIntersectingFeatures: aBoundingBox* is called, where *aBoundingBox* is the MBR of the AOI.

6. Removing a Feature from the Quadtree. Data deletion from a cell simply removes the data from the cell's *features* collection. The feature to be removed is retrieved from the quadtree using one of the methods described above, and then is deleted from the *features* collection. If the removal of the feature causes the *features* collection to become empty, the instantiated quad cell stays intact. Note that removal of features from a quadtree is a rare occurrence given the nature of geographic data.

7. Conclusions. In this paper, we have described various data types that are stored in the GIDB. Use of the MBR in our database allows for storage of these multiple data types within the same quadtree design. Based on the success of our GIDB prototype, we have found that a quadtree organization for spatial indexing is ideal for fast feature storage and retrieval in an object-oriented environment.

Acknowledgements. We would like to thank the Marine Corps Warfighting Lab, the Defense Modeling and Simulation Office, and the National Imagery and Mapping Agency's Terrain Modeling Program Office for sponsoring this work.

REFERENCES

- [1] A.B. CHAUDHRI AND M. LOOMIS, *Object Databases In Practice*, Chapter 15, New Jersey: Prentice Hall PTR, 1998, pp.234-253.
- [2] DEFENSE MAPPING AGENCY, *Military Standard: Vector Product Format. Draft Document No. MIL-STD-2407*, Fairfax, VA: Defense Mapping Agency, 1993.
- [3] M. COBB, M. CHUNG, K. SHAW, AND D. ARCTUR, *A Self-Adjusting Indexing Structure For Spatial Data*, GIS/LIS'95 Proceedings, Volume 1, 14-16 Nov 1995, pp.182-192.
- [4] M. CHUNG, M. COBB, K. SHAW, AND D. ARCTUR, *An Object-Oriented Approach For Handling Topology In VPF Products*, GIS/LIS'95 Proceedings, Volume 1, 14-16 Nov 1995, pp.163-174.
- [5] M.J. EGENHOEFER, *Spatial Query Languages*, Unpublished Ph.D. Thesis, University of Maine, 1989.
- [6] M. WORBOYS, *GIS: A Computing Perspective*, Taylor & Francis Inc., London, UK, 1995.
- [7] CLEMENTINI, E.J. SHARMA, AND M.J. EGENHOEFER, *Modelling Topological and Spatial Relations: Strategies for Query Processing*, Computers and Graphics, 18:6 1994, pp.815-22.
- [8] A. FRANK AND M. EGENHOEFER, *Object-Oriented Database Technology for GIS: Seminar Workbook*, in GIS/LIS'89, 1988.
- [9] M. COBB, M. CHUNG, K. SHAW, AND D. ARCTUR, *Object-Oriented Database Design and Implementation Issues for Object Vector Product Format*, NRL FR/7441-95-9641, Naval Research Laboratory, Stennis Space Center, 1996.

DESIGN OF A JAVA INTERFACE TO A SMALLTALK OO MAPPING DATABASE UTILIZING CORBA

RUTH WILSON
Naval Research Laboratory
Stennis Space Center, MS

TODD LOVITT
Planning Systems, Incorporated
Slidell, LA

MARIA COBB
University of Southern Mississippi
Hattiesburg, MS

MIYI CHUNG
Naval Research Laboratory
Stennis Space Center, MS

BARBARA RAY
Planning Systems, Incorporated
Slidell, LA

KEVIN SHAW
Naval Research Laboratory
Stennis Space Center, MS

ABSTRACT

The Digital Mapping, Charting and Geodesy Analysis Program (DMAP) of the Naval Research Laboratory has developed an object-oriented (OO) digital mapping database prototype, called the Geospatial Information Database (GIDB), capable of importing any of the National Imagery and Mapping Agency's Vector Product Format data and converting the data into an object format. The DMAP Team has also investigated existing OO technology that would allow the transfer and retrieval of data from the GIDB over the internet. This article describes how we have used a Java applet and the CORBA standard to succeed in our endeavor to make mapping data from our GIDB accessible over the World Wide Web.

KEY WORDS: Java, Smalltalk, CORBA, Object-Oriented, Mapping, Database

1 INTRODUCTION

The Digital Mapping, Charting and Geodesy Analysis Program (DMAP) of the Naval Research Laboratory began an effort in 1994 to develop an object-oriented (OO) digital mapping database [1]. The OO data model is derived from a relational data model known as Vector Product Format (VPF), a digital mapping standard produced by the National Imagery and Mapping Agency (NIMA) [2]. The need for the OO data model stems from the difficulty in representing complex geographic data by decomposing the data to fit the flat file structure imposed by the relational model. The OO prototype that has been developed, called the Geospatial Information Database (GIDB), is capable of importing any of NIMA's VPF products and converting the data into an object format for display, query, and updating purposes [3]. This system has been extended to include OO models for NIMA's two other families of digital products, Raster Product Format (RPF) and Text Product Standard (TPS).

In early 1997, the DMAP Team began to research existing OO technology that would allow the transfer and retrieval of data from the GIDB over the internet. The emergence of Java as the OO language of choice for internet applications led us to seek a way to use a Java Interface to access our GIDB which is written in Smalltalk. The solution was found in the Common Object Request Broker Architecture 2.0 (CORBA 2.0) standard, which allows for interoperability between different OO languages and across multiple platforms.

Background information on CORBA and Object Request Brokers (ORBs) is given in section 2. Section 3 gives an overview of our system design, and section 4 describes the Interface Definition Language (IDL) for our map features. This is followed by a summary of the available functions of our application on the web in section 5. The implications of our system to the mapping community, as well as a discussion of future work, are found in section 6.

2 CORBA

CORBA was developed by the Object Management Group (OMG), a consortium of software developers and users. In 1994, OMG released the CORBA 2.0 standard, which specifies a middleware architecture for distributed object communication [4]. CORBA is a nonproprietary, industry-supported standard which has experienced much growth in recent years. Several of its benefits include scalability, vendor independence through interoperability, language and platform transparency, and support for reuse.

The main component of the CORBA specification is the Object Request Broker (ORB). The ORB is responsible for intercepting a request, locating the object for handling the request, and invoking the correct method on that object. This often involves converting parameters from a common data type to a language-specific data type and vice versa (a process known as marshalling and unmarshalling), as well as returning results from the invoked method. Any two ORBs that are CORBA 2.0

compliant can provide communication between their application objects, regardless of programming language or platform.

In our development, we use Visigenic's Visibroker as our Java ORB, and GemStone's GemOrb as our Smalltalk ORB. These two vendor ORBs allow communication between applications via CORBA's Internet Inter-ORB Protocol (IIOP). The mechanism through which objects are accessed through the IIOP and between ORB vendors is with the interoperable object reference (IOR). The IOR, which is managed by the ORBs and is invisible to application programmers, includes the ORB's internal object reference, Internet host address, and port numbers. It is the use of the IOR that allows for vendor-independent interoperability.

The Interface Definition Language (IDL) is perhaps the most significant component of the CORBA standard. The IDL is a universal notation for software interfaces: it provides the common definitions through which objects in different programming languages can communicate. The syntax of IDL is very similar to C++, but objects defined in IDL can be implemented in any programming language that has CORBA bindings, such as Smalltalk, Java, C++, and Ada.

3 PROJECT OVERVIEW

The goal of the project was to have an internet Java-based mapping client which would provide display and query capabilities for a set of geographic objects, such as raster images and vector features [5]. These geographic objects would be retrieved from a Smalltalk GemStone OO database management system (OODBMS) acting as a server. Communication between the Java applet, which is embedded in an HTML document on a web page, and the Smalltalk application is accomplished using CORBA-compliant vendor ORBs. The use of Visibroker and GemOrb is completely transparent to anyone accessing the applet. Figure 1 shows the basic architecture of our system.

The retrieval of features from the server database is based on the concept of area of interest (AOI). The first

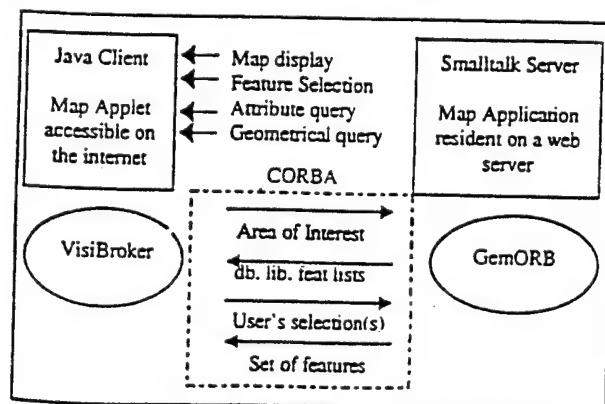


Figure 1. Basic system design.

screen of the applet displays a world map from which the user can select a location graphically through the use of a rectangle (bounding box). The user also has the option of entering the coordinates for the AOI manually. From the user input, a bounding box of the AOI is transmitted from the applet via CORBA to the Smalltalk server. The server responds with a set of database and library names for which data is available in that region. NIMA provides VPF data in databases, and each database contains one or more libraries. The user then selects a database and library, resulting in a list of coverages and feature classes being returned from the server through another CORBA request, as shown in figure 2.

Finally, the user selects the feature classes of interest and submits a request for them to be displayed. This request results in another CORBA communication, and the server returns a set of all of the features of the requested classes which are located in the given AOI. These features that are returned are complex objects with both geometric (coordinate) and attribute information. The applet can then display, select, and query on the returned features, as shown in figure 3. The available functions that have been implemented from within the applet will be discussed in detail in section 5.

4 INTERFACE DEFINITION LANGUAGE

The IDL is essential for communication between different programming languages. An IDL file must be created to allow for correct mappings of objects from one application to another; it is the means by which potential clients determine what operations are available and how they should be invoked. In our system, our IDL file defines all of the objects that are common to both client and server, as well as methods that can be invoked to perform certain operations on the server.

The first object that is utilized by both the Java client

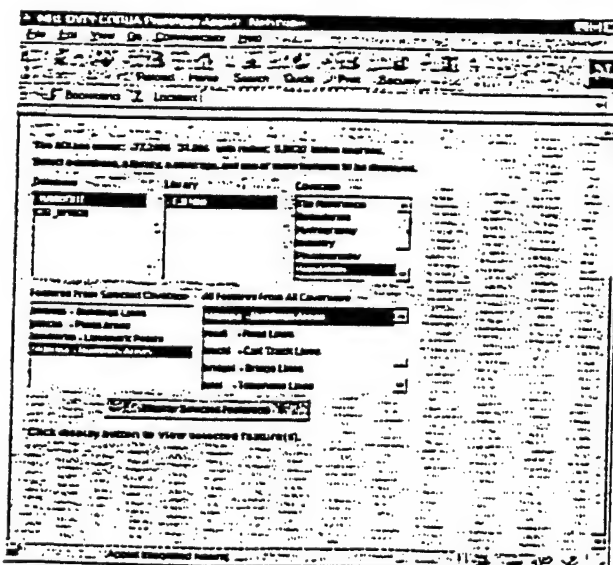


Figure 2. Database, library, and feature selection screen.

and the Smalltalk server is a bounding box for the AOI, an instance of the `VPFBoundingBox` class. To define our `VPFBoundingBox` object, we use a *struct* data type which allows related items to be grouped together. For example, *struct Point (float x,y);* defines a point to be made up of two float values, x and y. Similarly, our `VPFBoundingBox` is defined to be composed of two points, an origin and a corner: *struct VPFBoundingBox (Point origin; Point corner);*. We then defined an interface called *Integrated*, which contains the methods on the server that are invoked by the client. An interface is the most important aspect of IDL, since it provides all the information needed for a client to be able to interact with an object on the server. Shown below is our interface from the IDL file.

```
interface Integrated {
    StringCollection ReturnDatabasesForAOI(in
        VPFBoundingBox aBB);
    StringCollection ReturnCoveragesForAOI(in
        string dbName, in string libName);
    FeatureCollection ReturnFeatures(in
        StringCollection featColl, in
        VPFBoundingBox aBB, in string dbname, in
        string libname, in string covname);};
```

Note that the interface contains the method names with their parameters, as well as the data type of the returned object.

The most complex structure defined in our IDL is the struct `CORBAVPFFeature`.

```
struct CORBAVPFFeature {
    string featname;    string dbname;
    AttributeCollection attributes;
    Coordinates coords; long id;
    string covname;    string libname;
    VPFBoundingBox boundingBox;};
```

The Smalltalk application has an object class called `VPFFeature` from which our `CORBAVPFFeature` is

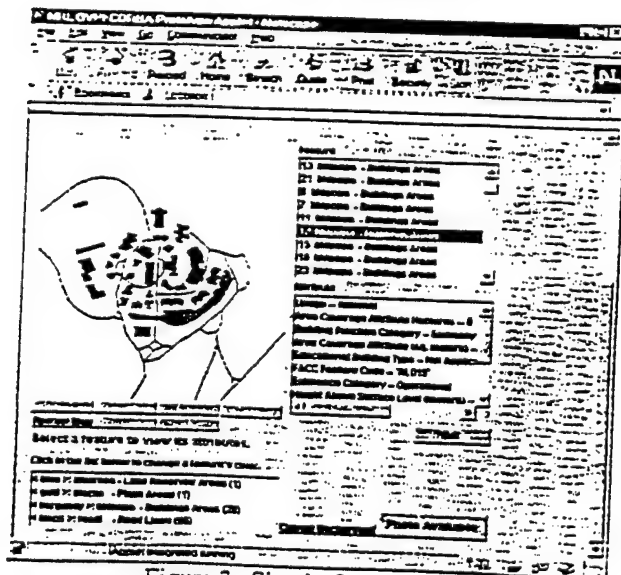


Figure 3. Simple Query screen.

derived. The Smalltalk `VPFFeature` class is more complex and has many more attributes, such as `featureDef`, `notes`, and `prims`. For our internet applet, though, only those attributes that are needed for display and user queries are defined as shown above.

Our IDL contains another structure which defines `CORBAAttribute`: *struct CORBAAttribute (string name; string value);*. The `CORBAAttribute` structure is used as part of the `CORBAVPFFeature` structure and gives attribute names and values for a given feature instance. For example, a given building may have an attribute name "Structure Shape of Roof" with attribute value "Gabled."

The final data type included in our IDL file is a *sequence*, which is similar to a one-dimensional array, but does not have a fixed length. We use sequences to reduce the number of messages passed from server to client. The size of each sequence is determined dynamically on both the server and the client. The following sequences are found in our IDL file.

```
typedef sequence<Point> Coordinates;
typedef sequence<string> StringCollection;
typedef sequence<CORBAVPFFeature>
    FeatureCollection;
```

This IDL file must be compiled on both the client and the server. On the server, the IDL is filed in, bindings to objects are made appropriately, and new methods are created. On the Java client, the process is similarly performed via an IDL to Java mapping. Objects defined in the IDL can then be referenced and used in both the client and server code.

5 FUNCTIONALITY ON THE WEB

The underlying motivation for having a web-based Java client access our OO mapping database is to give end users the ability to access and use NIMA data quickly and efficiently. At the present time, users of NIMA data must have software to view the data resident on their own computer systems, and must obtain the data on CD-ROM or other storage media. Our Java applet allows any user with a Java-enabled web browser, such as Netscape 4.0 or Internet Explorer 4.0, to access our GIDB over the internet and display NIMA map data available in their area of interest. In addition to display of map objects, we have extended the functionality of the Java client to include simple queries, individual feature selection, zoom capabilities, attribute queries, geometrical queries, and updates of attribute values. These functions were available in our stand-alone Smalltalk application, and have been adapted to our Java interface.

After the selected features in a user's AOI have been returned to the Java client and displayed, the user can change the colors of the features to distinguish between the feature classes retrieved. A color key is shown (figure 3) providing the color, feature class, and number of those

features in the given AOI. The user also has the ability to change the color of the background. Zoom capabilities are provided, allowing the user to zoom in, zoom out, or zoom to a user-specified area in the AOI. An individual feature may be selected by clicking on it in the map pane, resulting in the display of that feature's attributes.

A simple query is performed by clicking on the Query button below the map pane. This query lists all of the features in the map pane and gives the user access to each feature's attribute information, as shown in figure 3. More advanced queries can be performed by clicking on the Adv. Query button below the map pane. The advanced query screen allows users to display new feature classes in the AOI. The user can also perform attribute-level queries. For example, the user can request for all of the four-lane roads to be highlighted, or for all buildings that function as government buildings to be highlighted. Users can also perform geometrical queries, such as "find all buildings that are greater than 50 feet from the road," or "find all homes that are within 20 meters of the Embassy."

Update of feature attributes is also possible with the Java client. For example, a newly paved road could have its attribute for surface type updated from "gravel" to "concrete." This function of the applet must be password protected so that only users with authorization can change data in the database.

6 CONCLUSION

In this article we have discussed how our existing Smalltalk mapping application has been extended to the web utilizing a Java Interface via CORBA. The success of our effort is exhibited in the current functionalities of our Java applet on the web. We have several ongoing projects to improve our web application, including the display of NIMA's RPF and TPS data. We are currently

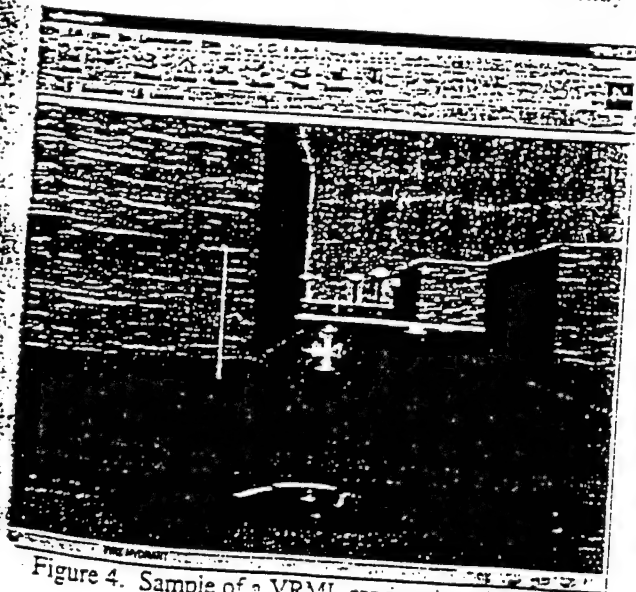


Figure 4. Sample of a VRML-rendered environment

in the final stages of our RPF display development. We are investigating ways to move to a truly distributed database. Additionally, we want to give users the ability to download data over the internet from our Java Interface in VPF format to expedite the distribution of NIMA data.

Another extension to our Java interface is the ability to display the features in our map pane in 3-D utilizing VRML 2.0. We anticipate the user being able to click on a "Render in 3-D" button to obtain a VRML generated 3-D model of the features in the current AOI. The open standard of VRML 2.0 is an excellent format for 3-D modeling of land and underwater terrain, natural features, and man-made features. We will generate 3-D models using gridded, TIN (Triangulated Irregular Network), and vector data. Our VRML models will provide additional information about the AOI by immersing the viewer into and allowing interaction with a virtual world (figure 4).

Once these tasks are accomplished, users interested in a wide variety of mapping data will be able to access and benefit from our GIDB over the internet from any platform using a Java-enabled web browser. This will allow the functionality of more powerful server machines to be exhibited on less capable client machines. It will also give users faster access to NIMA mapping data. Our migration to a Web-based mapping client is a revolutionary way of allowing clients with modest computing resources user-friendly access to state-of-the-art mapping data and software.

ACKNOWLEDGMENTS

We would like to thank the Marine Corps Warfighting Lab, the Defense Modeling and Simulation Office, and the National Imagery and Mapping Agency's Terrain Modeling Program Office for sponsoring this work.

REFERENCES

- [1] A.B. Chaudhri and M. Loomis. *Object Databases in Practice* Chapter 15, 234-253 (New Jersey: Prentice Hall PTR, 1998).
- [2] Defense Mapping Agency. *Military Standard: Vector Product Format. Draft Document No. MIL-STD-2407* (Fairfax, VA: Defense Mapping Agency, 1993).
- [3] K. Shaw, M. Cobb, M. Chung, and D. Arctur. Managing the US Navy's First OO Digital Mapping Project. *IEEE Computer*, 29(9), 1996, 69-74.
- [4] Object Management Group. *The Common Object Request Broker: Architecture and Specification*. (Object Management Group, Inc., July 1996).
- [5] M. Cobb, H. Foley, R. Wilson, M. Chung, and K. Shaw. An OO Database Migrates to the Web. *IEEE Software*, 15(3), 1998, 22-30.

GIS/LIS '98

ANNUAL CONFERENCE AND EXPOSITION

PROCEEDINGS

Sponsored by

AAG

ACSM

ASPRS

GITA

URISA



FORT WORTH, TEXAS

November 10-12, 1988

UNCERTAINTY ISSUES OF CONFLATION IN A DISTRIBUTED ENVIRONMENT

Maria A. Cobb
Assistant Professor
Department of Computer Science & Statistics
University of Southern Mississippi
Hattiesburg, MS 39406-5106
maria.cobb@usm.edu

Frederick E. Petry
Professor
Department of Electrical Engineering &
Computer Science
Tulane University
New Orleans, LA 70118
petrv@eecs.tulane.edu

Kevin B. Shaw
Digital Mapping, Charting & Geodesy Analysis Program
Team Leader
Mapping, Charting & Geodesy Branch
Naval Research Laboratory
Stennis Space Center, MS 39529
shaw@nrlssc.navy.mil

ABSTRACT

We have previously considered issues relative to conflation in a geographic information system (GIS) based on the Vector Product Format (VPF) developed by the National Imagery and Mapping Agency (NIMA) as the Department of Defense relational database interchange standard for vector mapping. In this context we developed a knowledge-based system to provide input on both spatial and non-spatial properties of geographic features. Degrees of matching were generated for candidate features based on measurable, objective measures as well as subjective ones. Matches for non-spatial properties (attributes) were generated on the basis of similarity tables developed for the allowed VPF attribute sets. These tables were used in conjunction with a fuzzy combination function to provide the overall degree of matching of the candidate features' attribute/value sets. An expert system also generated weights for the combination function using rules that represented semantic interrelationships of feature attributes.

We are currently developing a distributed object-oriented spatial database at the Naval Research Laboratory. Within this context, conflation is a very evident concern, as the distributed environment entails possible issues of schema (metadata) merging, as well as specific feature data conflation. Our previous approach using an expert system for conflation is not directly feasible in a distributed environment. Hence, we have developed a model of conflation tailored to a distributed environment, for which uncertainty issues handled by our previous conflation system are fully accounted.

INTRODUCTION

In recent years, the trend for most types of information systems has been a move to a more loosely coupled, distributed nature. The maturity of client-server architectures and software, as well as the virtual explosion of web-based systems, has demonstrated the enormous advantages of distributed systems. Furthermore, the advent of successful middleware technology such as the CORBA 2.0 standard and corresponding vendor implementations has drastically reduced barriers to communication and data sharing among heterogeneous software and hardware systems.

The interest in distributed geographic information systems (GIS) is no less than that of general information systems; however, the uniqueness of the nature of spatial data makes the issue of true

interoperability of GIS a major research concern. Evidence of this is abundant in the literature, and is also illustrated by initiatives such as the Open Geodata Interoperability Specification (OGIS) work by the Open GIS Consortium, Inc., as well as University Consortium on Geographic Information Science (UCGIS) priority research panels on "Interoperability of Geographic Information" and "Spatial Data Acquisition and Integration."

To set the context for our following work on conflation, we first define several of the most frequently used terms and their interrelationships within the general scope of GIS interoperability. These terms—*interoperability*, *integration*, *conflation* and *fusion*—are often used to convey very different ideas, or alternatively, used so loosely as to be somewhat interchangeable. Therefore, clarification of the use of these terms in this paper will be beneficial. Table 1 shows the 3-tier hierarchy illustrating our use of these terms.

At the lowest level of the hierarchy is the concept of data integration. In keeping with the most widespread use of this term, e.g. (Flowerdew, 1991), our use of data integration is intended to convey the idea of some process whereby incompatibilities among varying spatial data formats is resolved, allowing the various data types to be simultaneously analyzed/displayed/processed by a GIS. Data integration is therefore a low-level transformation procedure that requires no semantic knowledge of the various data. Integration of data types can be considered within the context of a single GIS, for example, the integration of vector and raster data for display purposes, or as part of a distributed system.

Conflation is a higher-level concept than integration, because it implies a deeper (semantic and "intelligent") knowledge about the data. Conflation results in a state of harmony among various data sources in which a single, "best" view of multiple data representations for similar data types is presented to the user. Thus, conflation logically can occur only if integration as defined earlier has already been resolved. Beyond conflation, which is viewed as an issue only among similar types of geographic information, e.g., vector with vector, the concept of data fusion is the more generic idea of combining widely varying forms of data, e.g., multimedia, in a system that can effectively organize the information in a way that is of benefit to the user. This concept of the "omni-informational" GIS is discussed in (Shepherd, 1991).

Finally, "interoperability" is viewed as the ultimate goal, encompassing all aspects of representation and semantic integration and providing a truly seamless view of geographic data in all its many forms. A UCGIS white paper (available at <http://www.ucgis.org/>) notes several long-term goals related to interoperability, including machine-interpreted semantics of geographic data, improved semantic representation for the data, language support for communication of geographic information and the development of canonical data models of geographic information.

Interoperability	Hierarchy of terms	Examples
	1. Fusion	Image + Text + Video + Vector + Raster + ...
	2. Conflation	Bridge representation 1 + Bridge representation 2 "Best" bridge representation
	3. Interchange	Proprietary vector format 1 Proprietary vector format 2

Table 1. Hierarchy and examples of terminology.

We view our work in conflation as being most relevant to the last of these goals, the development of a canonical data model. Though the issue is not specifically addressed in this paper, our approach to

conflation using a generic object model has obvious implications to issues of semantic schema integration and meaningful data interchange, as well as other research concerns in this area.

The following section gives pertinent background information on conflation research in general and within the context of our work. Following that is a discussion of aspects of uncertainty as related to feature matching in a distributed environment. The object conflation model is then presented in its general form, together with an example that is used to illustrate specific matching capabilities utilizing previously developed techniques for reasoning under uncertainty. Our summary and plans for future work are then given.

BACKGROUND AND RELATED WORK

Conflation

Conflation is typically regarded as the combination of information from two digital maps to produce a third map that is better than either of its component sources. The history of map conflation goes back to the early to mid-1980's. The first clear development and application of an automated conflation process occurred during a joint United States Geological Survey (USGS)-Bureau of the Census project designed to consolidate the agencies' respective digital map files of U.S. metropolitan areas (Saalfeld, 1988). The implementation of a computerized system for this task provided an essential foundation for much of the theory and many of the techniques used today. Since that time, others, including commercial GIS vendors, have implemented conflation tools within their applications. For an example of commercial work on conflation, see (Siegel, 1995).

Conflation can, in brief, be viewed as a multi-step, iterative process that involves feature matching, positional re-alignment of component maps and attribute deconfliction of positively identified feature matches. Feature matching, simply and perhaps somewhat obviously stated, involves the identification of features from different maps as being representations of the same geographic entity. Positional alignment is a mathematical procedure in which previously identified matching features are brought into spatial agreement, while deconfliction is a process in which contradictions in a matching pair's attributes and/or values are resolved. Positional alignment and deconfliction are both steps that are performed after a positive match has been determined. As such, it is easy to see that accurate feature matching results are essential to the overall quality of the resulting conflated map. Because of this dependency, our work thus far has concentrated solely on feature matching aspects of conflation. Our motivation for this work includes an effort to make individual geographic features "intelligent" enough to know when and how to conflate themselves in a distributed environment.

Original Conflation Project

Our original work on conflation (Cobb et al., 1998a) was performed within the context of the Digital Mapping, Charting & Geodesy Program (DMAP) at the Naval Research Laboratory (NRL), Stennis Space Center, Mississippi. DMAP was established in the 1980's as an avenue for providing comprehensive reviews of the National Imagery and Mapping Agency's (NIMA) newly developed digital mapping products. DMAP reviews of these products in the mid-1990's led to the recommendation of an object-oriented (OO) data model. Subsequently, DMAP received funds to develop an OO prototype for various NIMA products. This original prototype, Object Vector Product Format (OVPF), has since been significantly expanded and is now capable of processing various other vector data types, as well as raster and text data. The implementation also includes an OO database management system (ODBMS) integrated with an area-of-interest, web-based query model. Critical stages of the system development are documented in (Arctur et al., 1995; Shaw et al., 1996; Shaw, Chung and Cobb, 1998; and Cobb et al., 1998b).

(Cobb et al., 1998a) represents the first time that methods for reasoning under uncertainty have been utilized in a conflation process. The intent was to design a system that would mimic the reasoning process

of a human expert in conflation. A proof-of-concept system using NRL's previously developed OO prototype for spatial data (now known as the Geospatial Information Database, or GIDB), in conjunction with the Nexpert commercial expert system by Neuron Data, Inc., was implemented to demonstrate the effectiveness of the techniques for portions of attribute and shape similarity matching. More details of this system can be found in (Foley, 1997a-b).

The scope of the original project was NIMA's Vector Product Format (VPF) data (DMA, 1993). The VPF is a vector data standard that describes a general hierarchical model for storage and interchange of attributed vector data. Levels of the hierarchy include, top to bottom: *database*, which contains multiple *libraries*, each of which contains multiple *coverages*, each with multiple *feature classes*, each containing the actual geographic feature data. Within the VPF guidelines, various product types exist that adhere to the VPF standard, and add further restrictions to data content and format. A VPF database is defined by the product standard and a spatial extent; a library is defined by extent and scale, while a coverage is defined by thematically and topologically related features. Examples of VPF products include Digital Nautical Chart (DNC), Vector Smart Map (VMAP) and World Vector Shoreline Plus (WVS+). Each VPF product is designed to serve a particular user need.

The VPF standard provided a unifying framework for the conflation effort. VPF utilizes geospatial data standards such as the Feature and Attribute Coding Catalog (FACC) of the DIGEST specification (DGIWG, 1994). The FACC provides standard feature names for geospatial features, as well as attribute codes and encoded and non-encoded representations for attribute values. Commonalities such as this among the various product types allowed us to develop a conflation model that was valid for all databases that were VPF-compliant. Furthermore, though the implementation was specific to VPF, the principles of the model are generally applicable to other representations of attributed vector data.

As mentioned earlier, the focus of all our conflation research thus far has been feature matching, due to the dependence of other phases of conflation on correctly obtained results for matched features. In this paper, we extend the previously developed conflation model by developing a more general object-oriented approach and considering the model in a distributed environment.

UNCERTAINTY IN A DISTRIBUTED ENVIRONMENT

Uncertainty and Feature Matching

The assessment of feature match criteria is a process in which evidence must be evaluated and weighed and a conclusion drawn—not one in which equivalence can be unambiguously determined. For example, fuzzy concepts such as "closeness" of two features and "similarity" of attributes and feature groupings are essential for determining equivalence.

In fact, feature matching can be considered as a type of classification problem. That is, we are trying to determine whether one feature belongs to the same "class" as another; in this case the class is defined by a set of two features that are believed to be representations of the same real-world entity. This type of problem can be handled through theories of evidential reasoning or uncertainty, such as fuzzy logic (Zadeh, 1965) or Dempster-Shafer theory (Shafer, 1976). These theories attempt to provide likelihood measures for questions based on available, though not necessarily conclusive, evidence. For example, in feature matching, the question we must consider is, "Based on the available evidence, what is the likelihood (or probability) that feature A from map coverage 1 represents the same entity as feature B from map coverage 2?"

Our approach to feature matching draws from aspects of both fuzzy set logic and evidential reasoning. In particular, the assignment of matching scores for linguistic attributes is directly motivated by work in modeling linguistic variables by the use of fuzzy sets. For example, we need to be able to determine the semantic similarity of an attribute such as **road surface type** which may have a value of 'hard' for one feature and 'asphalt' for the potential matching feature. Obviously, the semantics of the two are not strictly equivalent, but neither are they contradictory. Likewise, the idea of combining scores from the different

components of feature matching to arrive at a single matching score is very similar to techniques for the combination of evidence used in evidential reasoning.

Uncertainty and Conflation for Distributed Data

Obviously, issues of uncertainty that apply to conflation within a single system—such as those for feature matching—are also applicable to conflation in a distributed environment. However, we believe additional factors related to the general topic of distributed databases increase the scope of uncertainty that must be considered in this context. As background, we can draw from the abundance of past and ongoing research in the realm of schema merging for conventional (i.e., relational) distributed heterogeneous databases. An example of work in this area includes (Lim, 1996).

The general concept of schema merging involves resolution of incompatibilities in metadata. These incompatibilities may be either *structural* or *semantic* in nature. Structural incompatibilities involve those, for example, in which attributes for representing the same values are defined differently. These may include different names for the attributes, or different domains for their associated values, e.g., float vs. integer. Semantic incompatibilities, on the other hand, represent those cases in which similarly defined attributes have different meanings or values. For example, an attribute of width for a road in one database may include the width of the road plus any associated right-of-ways, while the same attribute name in another database may only imply the width of the paved/driveable portion of the road. Semantic incompatibilities are much more difficult to handle automatically, as they necessarily imply a deeper understanding of the data.

It is clear that these issues are very similar to ones that must be faced in performing conflation in distributed spatial databases. In particular, semantic schema integration and the feature-matching phase of conflation require similar levels of knowledge regarding the meanings that various data are intended to convey. Semantic knowledge is inherently uncertain, as interpretations of even the most seemingly unambiguous words and phrases vary among individuals. Similarly, structural differences in spatial data representation for like features are to be expected in any distributed system comprised of heterogeneous data sources. It is evident from this discussion that conflation in a distributed environment can be viewed as a specific application of issues related to uncertainty in schema merging.

CONFLATION MODEL

As defined earlier in this paper, conflation is considered to be the process of combining information from two map sources to produce a better map. This definition, however, arises from the historical perspective of conflation in which a cartographer manually combines information from multiple paper maps. In the digital arena, in which data integration can virtually eliminate the concept of a standalone map, one of the first issues to be resolved for the development of a general conflation model is, "What constitutes a map source?"

From the perspective of the VPF standard, as well as similar vector data models, there are several obvious choices. Each of the levels in the VPF database hierarchy—database, library, coverage and feature class—could be considered as a candidate object for the role of defining a map source. However, to impose such a definition on a predefined subset of data, however seemingly natural a fit, would hinder our efforts in progressing to the ultimate goal of truly transparent spatial data integration and interoperability. Furthermore, conflation decisions made at any of these levels would have a possibly negative impact on fitness for use of the end result, by the inclusion or exclusion of data based on high-level properties. This is obviously undesirable, as fitness for use is an extremely significant qualitative measure of conflation success.

Definition of a map source is more generally related to the question of *when* to perform conflation. The following list gives examples of answers to this question:

- Automatically, whenever a user requests data

- Whenever data from two different overlapping databases, libraries, coverages, etc., are retrieved
- Only when explicitly requested by the user

The third possibility is obviously error-prone, as it assumes the user knows the cases in which conflation is an issue. The second preserves artificial categorizations of data that, as mentioned earlier, are barriers to long-term goals of data independence, integration and interoperability. Therefore, we have chosen the automatic system model represented by the first answer. In support of this, our concept of a map source is no longer some collection or sub-collection of geographical data; rather *each individual feature* is analyzed separately by the conflation system. More discussion on the implications of this follows in the presentation of the conflation model.

A second general consideration to be made is the impact of data distribution on a general conflation model. The issue in this case is whether to tailor the model to fit a particular distributed environment, or to design a "one-size-fits-all" model that effectively eliminates distribution as an issue. Although thorough consideration of partitioning schemes for spatial data is crucial for optimal performance in a given system, we have chosen to develop a conflation model at the logical level. Hence, in keeping with generally accepted principles of distributed (as well as non-distributed) database design, this model is completely independent of physical concerns such as actual data partitioning. However, the general issue of considering conflation within a distributed system versus a standalone system does impact design decisions, even at an abstract logical level. The primary issue that we consider is centralization versus distributed control of conflation events. A more in-depth discussion of this is given in conjunction with the presentation of the model.

Introduction to the Model

The presentation of our logical design is based on an OO model. The OO paradigm is well accepted as the prevailing method for the representation and manipulation of complex data such as geographical information. Within an OO framework, one is able to define models of real-world data in ways analogous to those in which we intuitively perceive and interpret those data. As a general introduction to the subject, an *object* is a collection of data (state) and methods (behavior) which represents the properties and processes of a real-world entity, such as the Chesapeake Bay Bridge or the Mississippi River. A *class* is a template for creating new objects that share common properties. For example, there may be a bridge class that captures generic information that every *instance* of that class (an instantiation) should contain. This would most likely include data such as length, height, maximum weight limit, and type (drawbridge, suspension, etc.). Examples of procedures for a bridge class could include opening and closing.

The packaging of an object's data with its procedures is known as *encapsulation*. Encapsulation allows modifications/additions to the system with minimal impact on other system components. This property is crucial for the successful development and maintenance of complex software systems such as GIS. Other major concepts for understanding OO models include *inheritance* and *polymorphism*. Inheritance is the automatic inclusion of attributes and methods from classes defined at a higher level in a class hierarchy. The class hierarchy is designed with more general classes being placed at higher levels and more specific classes being placed at lower levels. Inheritance reduces the need to duplicate data and code, while the structuring of a class hierarchy provides a logical organization of object "types." Polymorphism allows methods for different objects to have the same name, while providing different implementations. For example, both a circle and a square class could implement a method that calculates area. Both methods could be named *area*, but the implementations would use the appropriate formula for either a circle or a square. Methods that invoked the *area* method for an object would use one name--the class of the receiving object would determine which implementation to use. Polymorphism helps to simplify system design, and reduces coding complexity by eliminating error-prone if-then-else type-checking constructs.

These concepts are applied to our work in the following ways. Encapsulation allows each geographic feature to maintain its own state of knowledge related to information and procedures necessary for conflation with another feature. Changes in the algorithms/implementations of a particular feature's conflation abilities do not affect other features. Inheritance allows us to describe general types of conflation knowledge applicable to multiple feature classes within a single class at a high level in the

hierarchy. Lower-level (more specific) classes then automatically inherit this knowledge. Finally, polymorphism allows us to implement general conflation procedures that are valid for instances of any feature class; thus, polymorphism greatly reduces the need to be concerned with the issue of "type." The result is that we are able to treat, for example, railroad features and building features in the same manner at a logical level. Figure 1 shows a simplified class hierarchy for conflation.

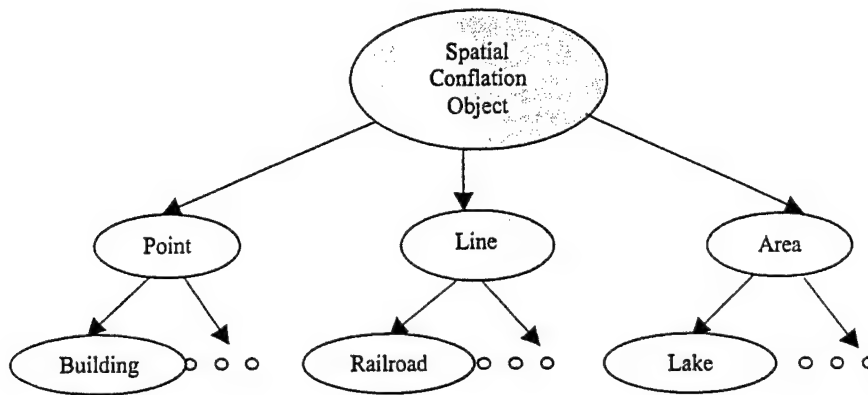


Figure 1. General class hierarchy for conflation.

Discussions and presentations at a recent assembly of the University Consortium of Geographic Information Science (UCGIS) lend credence to the primary points in this research. (White papers of the assembly are available at <http://www.ucgis.org/>.) First, the priority research panel on "Extensions to Geographic Representation" recommends the use of object-oriented techniques for improved representational power of geographic data. Second, the priority research panel on "Spatial Data Acquisition and Integration" states in their white paper that a "general theoretical and conceptual framework" for conflation is needed, and that furthermore, this framework should allow for matches of limited confidence (uncertainty).

Conflation Process Overview

The NRL GIDB prototype, within which proof-of-concept implementation of this model is currently being performed, is centered on the concept of spatial range queries, also known as area-of-interest (AOI) queries. Given an AOI, through definition of manual or graphical bounding box coordinates or geographical place name, the GIDB is able to return any vector, raster or multimedia data available for that area. Advanced queries, such as those limiting vector data attribute values, are also available.

Our conceptual model of the conflation process is based on this system model of AOI queries, limited of course to the consideration of vector data only. A diagram of the conflation process is shown in figure 2. The primary point to note is that the process takes place at the individual feature level, irrespective of that feature's physical residence, or logical database, library or coverage inclusion. In the first step, the user selects an AOI. The query manager then retrieves all feature objects from the distributed database that fall within the AOI (subject to any other constraints imposed by the user). From this collection of objects, the query manager randomly selects one object to send a "conflate" message. That object follows the protocol for determining which, if any, of the remaining objects in the query collection are matching representations for itself. Any object determined to be a match is placed in the matching feature set for the conflating object, ranked according to similarity scores. Objects that have been placed in a matching feature set are removed from the general query collection, so as not to be candidates for subsequent conflation iterations. This philosophy implies that only those objects that have scores *strongly* suggesting matching features are placed in the matching feature sets. The query manager continues by sending "conflate" messages to the remaining members of the query collection. When the process is complete, the query manager returns the

query collection for display. For any features with non-empty matching feature sets, the member of the set with the highest quality score (NOT the same as the similarity score) is returned as the representative for that feature.

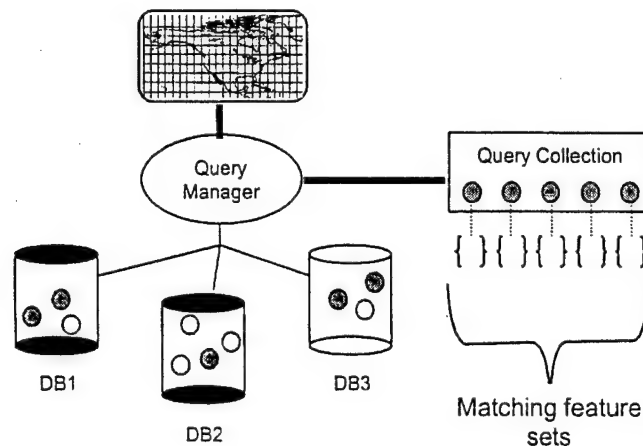


Figure 2. Conceptual model for distributed conflation.

Several points on the conceptual model are worth noting before we proceed with the details. First, the responsibility for conflation is distributed. Each individual feature contains the inherited knowledge needed for performing feature matching for that particular class of geographic objects. This is preferable to a centralized conflation system, where a single conflation object "manages" the process. Such a system would be more difficult both to implement and maintain due to differences in the ways in which objects from various feature classes should be matched, as well as the ramifications of adding new features and/or system nodes for a distributed system. Second, the process is automatic. Because the query manager collects the features satisfying the user query and invokes the conflation method for each object before returning the final set, the user does not have to explicitly request conflation, or even need to know that conflation is an issue. Third, when the results are returned to the user, only a single representation of each object is presented. For now, our treatment of deconflation is simply to select the "best" feature from a matching feature set, based on various objective and subjective criteria. This idea of simplifying the user's concern and involvement with conflation is critical for end-user based systems, as the whole need for such an automated approach is derived from the concept of non-expert users. Of course, this model could easily be extended to allow the user more involvement when desired.

The Spatial Conflation Object

The spatial conflation object (SCO) is shown in figure 1 as the top of the conflation hierarchy. As such, the SCO is, in OO terms, an abstract superclass. Abstract superclasses do not have concrete instances associated with them; rather, they provide a mechanism for the inheritance of generic traits that apply to each of the subclasses. The SCO, therefore, provides general information needed for any type of feature conflation, as well as default actions to take during the process. This knowledge is not expected to be complete enough to allow for a well-defined conflation procedure. Instead, this knowledge is augmented with specific types of knowledge for feature class specializations. For example, the way in which geometric similarity is determined for line features vs. area features is different. The common trait is that the general idea of geometric similarity is a significant factor in performing conflation. To go one level further, the way in which geometric similarity is assessed for railroad lines vs. river lines is different, though the method for computing the measurement is the same, since both are line features. To summarize, the general knowledge for conflation is inherited through the class hierarchy; at each level, additional, more concrete knowledge is added until there is sufficient for successfully performing conflation.

Figure 3 illustrates the set of attributes and methods for the SCO. The attributes are the top, italicized set, while the methods are the lower, non-italicized set. This is not a complete set, but it is sufficient for

explanatory purposes. Two types of attributes are given, instance and class. These differ in that for instance attributes, each object instantiation of feature subclasses in the SCO hierarchy inherits this set of attributes, with each object having different values for each attribute. For class variables, there is one value, which is accessible to all instantiations of that class and its subclasses. Likewise, the methods are also inherited, although it is likely that these will often be overridden by feature class-specific methods. The ones defined at this level may be used as defaults, if no further information is available.

The *rank* attribute represents a measure of quality of the information for a particular feature. It is a subjective measure based on quality information provided in the features' database and library, as well as scale and fitness for use as determined by the user's objectives. Determination of this value is one of the major places in the model in which uncertainty must be taken into account. The *matchingSet* contains all features that were determined to be matching features for the object beyond a user-defined or default *threshold*. The *RuleSet* is a composite object comprised of rules and rule-processing strategies for measuring similarities in distance, geometry, attributes, topology, FACC feature codes, and other miscellaneous criteria for determining feature matches. Obviously, this is another area in which techniques for reasoning under uncertainty, as well as techniques for subjectively evaluating and analyzing vague, qualitative information must be used. The role of *SimilarityTableSet* is explained in a subsequent section.

As the method names are somewhat self-explanatory, we will only expound here on the way in which these are used in the conflation process. The *conflate* method is invoked on each object by the query manager, and sets in motion a sequence of actions. We assume for now, for the sake of simplicity, that each object's rank has been predetermined. The object then begins the process to determine if any of the other objects in the query set are potential matches (*findCandidates*). This process is one of filtering and refining. That is, simple measures that can quickly eliminate non-matches are used first, e.g., completely different feature codes, followed successively by more subtle checks such as attribute set and value similarity. Once a candidate has been evaluated, it is determined whether the overall *matchingScore*, as determined by a function combining *attributeScore*, *geometryScore*, *topologyScore* and *filterScore*, exceeds the *threshold* level. Those that do are placed into the *matchingSet*. The best of these features, as determined by the *rank*, is returned to the query manager as the representative for that feature (*selectFeature*).

Spatial Conflation Object

<u>Instance</u>	<u>Class</u>
<i>rank</i>	<i>RuleSet</i>
<i>matchingSet</i>	<i>SimilarityTableSet</i>
<i>threshold</i>	
<i>attributeScore</i>	
<i>geometryScore</i>	
<i>topologyScore</i>	
<i>filterScore</i>	
<i>matchingScore</i>	
<hr/>	
<i>conflate</i>	
<i>computeRank</i>	
<i>findCandidates</i>	

Figure 3. Attributes and methods for the SCO.

INTEGRATION OF MONOLITHIC CONFLATION MODEL

Previously, we have developed attribute and shape similarity measures for conflation based on principles of fuzzy logic and evidential reasoning (Foley, 1997a-b; Cobb, et al., 1998a). The implementation of the resulting methodology involved a three-tier architecture consisting of a Smalltalk component for the user and OO database interface, a commercial expert system for rule processing and a C

language interface between the two. This setup worked well for a proof-of-concept implementation of a single system/database; however, with the current emphasis on distributed spatial databases, it is obvious that this initial architecture is too cumbersome to extend to a distributed realm. Therefore, in this section we show how the aforementioned similarity measures can be migrated to and utilized in the distributed object conflation model given in this paper.

Attribute Matching Algorithm

For attribute matching, each feature object is considered as a set of attribute-value pairs:

$$((a_{11}, v_{11}), (a_{12}, v_{12}), \dots, (a_{1n}, v_{1n}))$$

$$((a_{21}, v_{21}), (a_{22}, v_{22}), \dots, (a_{2m}, v_{2m}))$$

We consider matching for the two categories of numeric and linguistic attribute domains. In general, matching for numeric domains is handled through the use of membership matching functions, while matching for linguistic domains is handled through the use of attribute similarity tables. For the purpose of example, we will consider here only the linguistic domain.

A similarity table for a specific attribute contains a value in the range [0,1] for each attribute domain value. Each of these values represents a degree of matching between two attribute values. In many cases, the domain values are integers that represent encodings of linguistic characteristics; thus, the similarity values in the table represent similarity between linguistic terms. Matching for features based upon attribute similarity is a two-phase process. First, the similarity between each of the attribute values for the two features is determined from a similarity table. Second, measures of semantic interrelationships between and among the various attribute values are computed within a rule-based expert system environment. Based on these interrelationships, the expert system returns one or more weights for increasing or decreasing the matching score for various attributes.

As an example, consider a railroad feature with an attribute RRA, representing the railroad power source, such that the attribute can have the following values.

0	Unknown	4	Non-electrified
1	Electrified Track	999	Other
3	Overhead Electrified		

The similarity table for RRA is given in table 2.

RRA	0	1	3	4	999
0	.2				
1	.2	1			
3	.2	.6	1		
4	.2	.1	.1	1	
999	.2	.2	.2	.2	.2

Table 2. Similarity table for railroad power source.

An example of a production rule for two railroad features, RR1 and RR2, is:

IF ((RR1.ltn = 3 and RR2.ltn = 2) and (RR1.rrc = 16 and RR2.rrc = 16))
THEN $w_{rra} \leftarrow 1.0$ and $w_{ltn} \leftarrow 0.5$

where ltn represents the number of tracks and rrc represents railroad categories. The overall matching score for attributes is given by:

$$MS_{ij} = (\sum_{k=1, N} [simA_k(F_i, F_j) \times ESW_{A_k}]) / N$$

where A_k is the k^{th} attribute in both F_i and F_j ; N is the number of attributes that are common to both F_i and F_j , and ESW is the weight computed by the expert system. Explanations of the derivation of similarity values and semantic production rules would require more depth than space will permit here; the reader is thus referred to the previously mentioned references on this work for greater details.

Attribute Matching and the Distributed Model

We begin by summarizing the contents of the previous section for illustrating the application of the distributed model to this work. The matching of attributes for linguistic-based value domains involves: (1) a *similarity table* for determining matching between two attribute values; (2) *production rules* for considering semantic implications of two or more attribute values; (3) *weights* generated by these rules, which are combined with previously determined matching scores; and (4) an *attribute matching score* computed over the set of common attributes for two features.

Beginning with the first component in this list, it is relatively easy to see that the idea of similarity tables for linguistic attribute matching applies across all feature classes and representations. Therefore, an attribute for this is explicitly provided in the SCO (see figure 3). The *SimilarityTableSet* attribute is shown as a class variable. This variable is inherited by all feature classes, e.g., bridges, roads, etc., each of which has a unique set of tables, one per attribute of that class, that contains values representative of the attribute value domain. For example, the similarity table for RRA shown as table 2 would be a member of the *SimilarityTableSet* for a railroad feature class. Because it is a class variable, only one copy is maintained that is accessible to all instances of that class.

The incorporation of production rules into the model is a complex issue, of which only the very basics will be considered here for illustrative purposes. As a glimpse into the nature of this complexity, we note that further refinement of the SCO model involves the modeling of a rule-based production system as an object that includes all the behavior necessary for processing rules of varying formats and combinations, as well as including all the data needed by the rules. As such, production rules obviously are one aspect of the model that are again applicable across feature classes and representations. The generic model, or template, for production rules is thus applicable at the SCO level. However, specific instantiations of rules must incorporate data and knowledge from lower levels, including both the feature class level (this is illustrated in the rule example given previously for railroad attributes), as well as the second level of the hierarchy shown in figure 1 for point, line and area classes.

The weights generated by the production rule system are intermediate data, and are thus considered as part of the object model for that system. No knowledge of the values of these weights is required of the individual features involved in conflation. The attribute matching score, as well as matching scores for the various other categories of matching criteria, are modeled as instance variables that are inherited by the feature classes (figure 3). Each geographic feature maintains knowledge of these intermediate scores to compute a final matching score (*matchingScore*). The intermediate scores are maintained so that knowledgeable users can view the results as part of a possible conflation verification procedure. The equation given for computing the attribute matching score is valid across feature classes and thus belongs in the SCO. However, knowledge for interpreting the results is conceivably different across feature classes, and thus is maintained at the lower level of the model hierarchy.

Allocation Issues and the Distributed Model

Though irrelevant at a conceptual level, physical allocation of data in a distributed system is of paramount concern for performance issues. Here, we briefly consider one possible allocation scheme and the potential impact as related to the SCO model.

The scheme we consider is a partitioning based on representational classes. That is, all line features such as railroad lines, power lines, etc., reside on the same physical node; all point features are likewise allocated to the same node, as are all area features. In consideration of the hierarchy given in figure 1 within this setup, we believe the optimal partitioning of the hierarchy is to include all level 3 (feature class) classes and their instantiations together with their corresponding level 2 superclass. For example, all transportation lines, utility lines, etc. would be co-resident with the line class object. For conflation, this setup would minimize network traffic needed for transfer of data and execution of methods, as most of the specific conflation knowledge is inherited from the SCO and values

instantiated at these lower two levels. Of course, other partitionings may also provide acceptable performance, but this one is given as an example of considerations between data allocation methods and the distributed conflation model.

SUMMARY AND FUTURE WORK

The conflation model presented in this paper, together with the discussion on attribute matching, illustrates how the various pieces of data and processes related to uncertainty are distributed throughout the various levels of an object hierarchy. The inheritance and encapsulation of this knowledge enables conflation in a distributed environment to take place transparently to the user, and without the need for a centralized conflation "manager." We believe the use of OO techniques for the development of a general conflation model is crucial to enabling complex conflation in a distributed environment. The use of an approach such as this has far-reaching implications and can tremendously impact progress in the crucial area of spatial data interoperability. For example, implementation of this approach could enable DoD and civilian mapping applications to seamlessly utilize NIMA, USGS, etc., holdings effectively.

Of course, further refinement of the model is needed to enable the handling of conflation for all feature types. Expansion must be made in the geometric and topological matching portions of the model. One issue related to the implementation of the model is the process of knowledge acquisition for the uncertainty-based components. Interaction with cartographic experts in the field of conflation is a necessary step in providing design and implementation-level details for the rule-based component of the model.

Another issue upon which we will be expanding is that of deconfliction. Whereas, for now we simply select the feature with the best overall quality, we are planning for the development of a process that will allow combinations of the best parts of matching features. Optimistically, this will provide a "super" feature that is better in all or most respects than any of the single matching features. Finally, the idea of a parallel conflation algorithm is an intriguing concept for future work within a distributed environment. Initial implementation of the distributed conflation model is currently underway at the Naval Research Laboratory, and we anticipate reporting on preliminary results in the near future.

ACKNOWLEDGMENTS

This work was sponsored by the Marine Corps Warfighting Laboratory (MCWL) under Program Element 0603640M, with Lt. Col. Bott as the program manager. The views and conclusions contained in this paper are those of the authors and should not be considered as representing those of MCWL.

REFERENCES

Arctur D., E. Anwar, J. Alexander, S. Chakravarthy, M. Chung, M. Cobb and K. Shaw (1995). Comparison and benchmarks for import of vector product format (VPF) geographic data from object-oriented and relational database files. *Proceedings of the 4th Symposium on Spatial Databases (SSD '95)*, Springer-Verlag, New York, pp. 368-384.

Cobb, M., M. Chung, V. Miller, H. Foley III, F. Petry and K. Shaw (1998a). A rule-based approach for the conflation of attributed vector data. *GeoInformatica*, 2(1):7-35.

Cobb, M., H. Foley III, R. Wilson, M. Chung and K. Shaw (1998b). An OO database migrates to the web. *IEEE Software*, 15(3):22-30.

Defense Mapping Agency (DMA) (1993). Military Standard: Vector Product Format, Draft Document No. MIL-STD-2407, Defense Mapping Agency, Fairfax, VA.

Digital Geographic Information Working Group (DGIWG) (1994). The Digital Geographic Information Exchange Standard (DIGEST), Directorate of Geographic Operations, Department of National Defense, Canada.

Flowerdew, R. (1991). Spatial data integration. *Geographical Information Systems: Principles and Applications*, eds. D.J. Maguire, M.F. Goodchild and D.W. Rhind, Longman Scientific and Technical, Great Britain, vol. 1, pp. 375-387.

Foley III, H., F. Petry, M. Cobb and K. Shaw (1997a). Utilization of an expert system for the analysis of semantic characteristics for improved conflation in geographic information systems. *Proceedings of the 10th International Conference On Industrial and Engineering Applications of AI*, Atlanta, GA, pp. 267-275.

Foley III, H., F. Petry, M. Cobb and K. Shaw (1997b). Using semantic constraints for improved conflation in spatial databases. *Proceedings of the 7th International Fuzzy Systems Association World Congress*, Prague, pp. 193-197.

Lim, E., J. Srivastava and S. Shekar (1996). An evidential reasoning approach to attribute value conflict resolution in database integration. *IEEE Transactions on Knowledge and Data Engineering*, 8(5):707-723.

Saalfeld, A. (1988). Conflation: automated map compilation. *International Journal of GIS*, 2(3):217-228.

Shafer, G. (1976). *A Mathematical Theory of Evidence*, Princeton University Press.

Shaw, K., M. Chung and M. Cobb (1998). Migration process and considerations for the object-oriented vector product format to objectstore database management system. *Object Databases in Practice*, eds. M.E.S. Loomis and A.B. Chaudhri, Prentice-Hall, pp. 234-253.

Shaw, K., M. Cobb, M. Chung and D. Arctur (1996). Managing the Navy's first object-oriented digital mapping project. *IEEE Computer*, 2(9):69-74.

Shepherd, I.D.H. (1991). Information integration and GIS. *Geographical Information Systems: Principles and Applications*, eds. D.J. Maguire, M.F. Goodchild and D.W. Rhind, Longman Scientific and Technical, Great Britain, vol. 1, pp. 337-360.

Siegel, D. (1995). A non-traditional solution to conflation. *Proceedings of the Urban and Regional Information Systems Association annual conference*, Washington, D.C., 16-20 July, pp. 462-75.

Zadeh, L.A. (1965). Fuzzy sets. *Information and Control*, pp. 338-353.

Migration Process and Consideration for the Object-Oriented Vector Product Format to ObjectStore Database Management System

*Kevin Shaw, Naval Research Laboratory
Miyi Chung, Naval Research Laboratory
Maria Cobb, Naval Research Laboratory*

ABSTRACT

This chapter presents an object-oriented approach for handling Vector Product Format (VPF) mapping databases as produced by the U.S. Defense Mapping Agency. This approach is implemented in the Object Vector Product Format (OVPF) Smalltalk prototype developed by the U.S. Naval Research Laboratory and the University of Florida. OVPF provides an integrated framework for four VPF products: Digital Nautical Chart, World Vector Shoreline Plus, Urban Vector Smart Map, and Vector Smart Map level 0. Having four VPF products and with the update capability, persistent storage of these spatial data was one of the concerns. This chapter will introduce the changes to the data model to accommodate the ObjectStore implementation and migration process of each of the products to the ObjectStore ODBMS.

We wish to thank our sponsor, DMA, Mr. Jim Kraus, and Mr. Jake Garrison, program managers, for sponsoring this research. We would also like to thank Mr. Mike Harris for performing the technical review of this chapter.

INTRODUCTION

This chapter documents the integration of the commercial object-oriented database management system (ODBMS) ObjectStore (by Object Design, Inc.) with the Object Vector Product Format (OVPF) Smalltalk prototype, including both design and initial implementation. The work documented in this chapter was performed as part of the Object-Oriented Database Exploitation within the Global Geospatial Information & Services (GGI&S) Data Warehouse project conducted by the Naval Research Laboratory and the University of Florida and sponsored by the Defense Mapping Agency (DMA). The purpose of the project was to determine the potential impact of object-oriented (OO) technology on DMA's GGI&S initiative. Other reports [Sha+95, Chu+95a, Arc+95a, Arc+95b] have documented the integration of multiple Vector Product Format (VPF) products into OVPF, network investigation results, and evaluation of a hybrid object-relational database management system.

As a brief orientation for the reader, VPF is a military standard for the storage and exchange of digital vector data, which consists of a hierarchical system of ASCII data files organized in third-normal relational form. It emerged in the late 1980s as DMA began converting its paper maps into digital format. The VPF relational data model, however, has problems representing complex spatial data. As a result, in 1991, the U.S. Navy, a DMA database user, began investigating how object technology could improve these digital maps. This research led to the development of the OVPF, an object-oriented approach to viewing and editing digital maps and charts. By combining multiple relational databases into a single OO database, OVPF offers users such key advantages as the ability to immediately update and modify the content of the original data. The initial OVPF prototype consisted of approximately 400 classes (in addition to those provided in the Smalltalk class library) and several thousand methods. Benchmarking and comparisons to VPF are documented in [Arc+95c] and showed roughly an order of magnitude improvement in importing time.

The OVPF prototype application has been designed and implemented with the ParPlace Systems' VisualWorks version of the Smalltalk programming language. This choice is primarily due to the sophistication and productivity of the Smalltalk development environment for building complex applications. Due to the semantic differences among OO programming languages, the terms and definitions used here may not apply consistently across all such languages. However, the concepts represented by these terms should be general enough to be implemented in other OO languages. The commercial ODBMS has been chosen with this in mind as well—ObjectStore includes interfaces to Smalltalk and C++ so that objects created and stored by a Smalltalk program should be accessible to programs written in C++. This is not yet possible with current products as we write, but industry efforts to develop cross-language compatibility for ODBMSs are underway.

In this chapter, design issues are emphasized with respect to OVPF, as well as the experiences incurred, lessons learned, and potential impact from the actual im-

plementation of ObjectStore, one of the leading commercial ODBMSs. The next section provides an explanation of why ODBMS is used along with an introduction to basic ODBMS concepts. The following section describes background information on object-oriented symbology and terminology, along with a brief description of the OVPF structure, including the design of the class hierarchies for importing and representing multiple VPF products. Implementation design to integrate ObjectStore with OVPF is presented in the next section, and the last section concludes with key findings and summary of the integration effort.

ODBMS INTEGRATION

Why Use an ODBMS?

With support for multiple VPF products now integrated into OVPF's framework, we turn to the issues and tradeoffs of support for commercial ODBMSs. This section presents our current experience and understanding with respect to the issues and effects of alternative ODBMS architectures on OVPF's design. In this regard, we will be distinguishing between internal and external databases. The internal database refers to OVPF's computer-memory-resident object space of metadata and feature objects. The external database refers to the disk-resident database implemented using the commercial vendors' ODBMS products as an alternative to the data storage.

In today's distributed processing environment, each OVPF user is likely to be working on a networked computer workstation that is separate from, but perhaps just as powerful as, the workstation housing the shared external databases. OVPF will be seen as a *client* process making requests for data from a *server* process running on the central host. Commercially available ODBMSs typically fall into one or both of two types of client-server architecture: object-server or page-server. The distinction is based on the unit of data transfer between the server and client processes. In an *object-server* architecture, the server understands the client's concept of an object. In this model, each transfer from the server to a client process is based on groupings of interconnected objects. A *page server*, on the other hand, is unaware of "objects" as such, but transfers data in units of a disk page that is typically 4 Kb.

As described above, OVPF can function without a database management system. Relational tables as stored in directories according to the VPF specification [DMA92] are processed and information is brought into memory upon import of one or more coverages. Once in memory, disk resident data are never subsequently accessed. The advantage to this approach, besides its simplicity, is that the memory resident data are quickly accessible for manipulation, eliminating the need to perform costly disk accesses and table joins. The disadvantages are primarily: (1) the amount of data that can be imported for a single session is limited by the capacity of

15 Migration Process and Consideration for Product Format

physical memory, and (2) data are not made available for concurrent access by multiple users; thus, changes to the data made through the use of OVPF are not readily apparent to others.

The use of an ODBMS eliminates both of these concerns, as well as providing additional advantages. For example, with this approach, OVPF is no longer limited by memory size for data import and viewing; data are simply stored in the database until needed, then brought into memory for display or editing purposes. Additionally, geographic object level security and auditing can be readily managed.

The function of any DBMS is to provide persistent (maintained from session to session) storage of data, controlled access to the data, and backup and recovery capabilities, among others. Object-oriented DBMSs provide these functions specifically for *objects*—units of data defined and assigned values through the use of an OO programming language such as Smalltalk or C++. While objects are generally considered to consist of both state (data) and behavior (procedures), ODBMSs are typically concerned only with the storage of the state information, as are traditional relational database management systems.

ODBMS Concepts

Following is a list of three significant high-level concepts concerning ODBMSs. Each is elaborated upon in the discussion that follows.

- Persistent versus transient objects
- Transactions and concurrency control
- Security and authorization

The distinction between persistent and transient objects is somewhat less clear for ODBMSs than for RDBMSs, due to the tightly coupled nature of the database with the application. *Transient objects* are defined to be those objects that exist in the computer's memory only during execution of the application, such as the window system objects for displaying the key data. *Persistent objects*, on the other hand, are those representing the key data, whose state is maintained in the database and exist even when there is no application program running.

Transient and persistent objects can coexist within a running process. The distinction becomes blurred because persistent data accessed from the database can be assigned to a transient object. It is important for application programs to manage both transient and persistent data in consistent ways so updates to persistent data are made when needed and so data that should remain transient are not inadvertently made persistent. For example, a web of interconnected transient objects can be made persistent simply by allocating space in the database for the "root object" of the web. Once this transaction is completed, the ODBMS will migrate the transitive closure (entire web) from the root object into the database. It is important, therefore, to be sure that such a transitive closure does *not* include references to ob-

ts, such as the window system in which the data are being displayed, as this would "pull" a very large and unnecessary part of the application objects into the database.

All accesses to persistent data are typically made within a *transaction*. A transaction is defined to be a sequence of instructions that access the database and whose execution is guaranteed to be atomic; that is, either all or none of the instructions are executed. *Read* transactions are defined as those that retrieve data only, while *write* (or *read-write*) transactions are those that actually change the data. When a transaction has successfully terminated, it is *committed*, meaning that any changes made to persistent data within the transaction are written to the database and will not be undone. If a transaction is *aborted*, the ODBMS will cause a rollback of changes made to the objects to return them to their state as existed prior to the beginning of the transaction.

A related issue is that of *concurrency control*. Concurrency control is an issue only for multiuser DBMSs and is concerned with ensuring, usually through the use of locks, that when a user is accessing persistent data within a transaction, other users cannot simultaneously change the data, resulting in the database being left in an inconsistent state. Database inconsistency can easily result from the interleaving of different users' instructions regarding the same persistent data.

It is widely recognized that data are one of the most valuable assets of any organization. With that recognition comes the need to protect such data. As a result, DBMSs typically provide some system of restricting access to data based on a user authorization system. Different sets of privileges may be provided for different classes of users. Privileges may include the right to read or modify data, as well as the ability to modify the database schema itself.

ObjectStore

ObjectStore was selected for integration with OVPF for the following reasons:

- It is a well-established, commercially successful ODBMS.
- It is available with both Smalltalk and C++ language interfaces.

ObjectStore ODBMS functions as a simple repository of data. Three major processes are used in ObjectStore: the server, the client, and the cache manager. These are shown in Figure 15-1. The database and the files consist of the cache, communication segment, and the database. The *ObjectStore server* provides I/O services to databases that reside on local and remote disks.

ObjectStore is based on a page-server architecture in which the server uses a two-phase commit protocol in conjunction with other servers to guarantee consistent transaction completion in a distributed database environment. The *ObjectStore client* allocates and deallocates storage for persistent objects. Communication with the ObjectStore server is managed for fetching and locking pages. Pages are

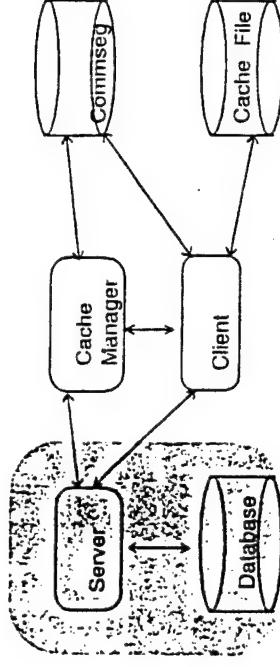


Figure 15-1 ObjectStore configuration.

mapped into the virtual address space of the client processes. Recently referenced pages are maintained and coordinated with the cache manager to hold locks as long as possible for minimizing network I/O traffic with the server. Transaction commits are performed to transmit any modification to the database. The *ObjectStore cache manager* handles asynchronous lock callback requests from the ObjectStore server. This capability provides immediate response to the ObjectStore server, which in turn leaves the ObjectStore client library free to make only synchronous requests.

To reduce network traffic with the server and to provide coordination via shared access to the client commseg file, a "lazy" lock release mechanism is coordinated with the ObjectStore client. The ObjectStore database can be accessed only by the server. However, the client is responsible for database organization and structure. Cache and commseg files are used for controlling encached data. The cache file is used as a swap space for persistent data, and as a backup store for in-memory persistent objects. The commseg file tracks the status of encached pages. This file is used by the client to determine whether a page can be accessed and locked. The cache manager uses this file to check and/or give up locks upon the server's request.

ObjectStore uses a "pessimistic" approach to transaction management that requires every database access to occur within the bounds of a transaction. This guarantees data integrity among all users at the cost of potentially blocking some users while one is accessing a particular page of data. Transaction bounds are explicitly stated by the use of messages to ObjectStore's class *OSTransaction*. This helps ensure that each transaction will be as short as possible, thus minimizing blocking of other users.

For a more detailed explanation of ObjectStore features, the reader is referred to [Sha+95].

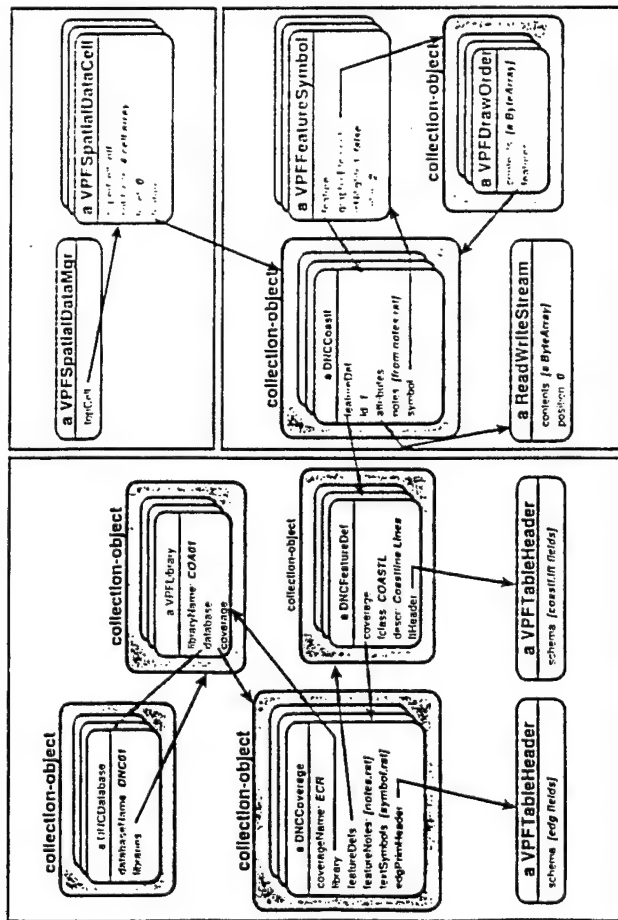


Figure 15-4 Partial feature data and metadata for DNC COASTL features.

Note the use of *collection-objects* in Figure 15-4—these are instances of predefined classes in Smalltalk that are used to encapsulate a collection of one or more types of other objects. Collection objects simplify the management of large groups of objects when it is important to be able to sequentially or randomly access the group members. Libraries, coverages, featureDefs, and feature objects are frequently held and accessed from collection objects in OVPF.

Metadata Classes and Instances

It will be helpful now to introduce the metadata (schema definition) classes in OVPF. VPFFeatureDef and its subclasses form an important link between feature objects and their defining schemas. Figures 15-2, 15-3a, and 15-4a show the classes that define the structure of each VPF database product. The classes VPFFeatureDef, VPFFeatureDef, and VPFFeatureDef all have product-specific subclasses. The need for product-specific subclasses is most concerned with the feature import process and will be discussed in the next section. What is more pertinent for now is that much of the feature metadata is stored in instance variables whose names end in "Header," such as *VPFFeatureDef*. These variables each hold onto an instance of VPFFeatureDef, which in turn holds a schema of information such

15 Migration Process and Consideration for Product Format

as the field id, description, data type, and length of each column in the table, as documented in the VPF product specification. Another role of the VPFFeatureDef schema is to support the actual conversion of source data bytes into usable numerical values and vice versa, as needed.

Feature Attributes

As shown in Figure 15-2, each subclass of VPFFeature inherits an *attribute* instance variable. The *attribute* variable holds onto the contents of a record from the *.VPF, *.LPT, *.ATF, or *.TTF feature table. Each of these records contains the integer or character values of attributes common to all features of a given class such as ACC (accuracy code) or SLT (shoreline type).

Figure 15-2c shows that VPFFeature also defines a *textAttributes* instance variable. This variable holds onto a set of values from the SYMBOL. RA file (if present) that defines text font, style (italic, bold, etc.), size, and color for one or more text features.

Both *attributes* and *textAttributes* are stored and processed in the same manner. They use a predefined Smalltalk class called *ReadWriteStream* that provides an efficient means of storing, reading, and updating an arbitrary sequence of bytes. Since floating-point and integer values are stored somewhat differently among the various computer systems (UNIX, DOS, and Macintosh), we chose to defer translating the bytes into user-understandable values until the user actually requests to examine them. In addition to reducing processing time during feature import, this also greatly reduces the total number of objects we have to handle. Since we are using a single *ReadWriteStream* object for each set of attributes instead of creating a separate integer or character value object for every attribute, we are greatly reducing the load on the computer system's internal I/O channel and memory.

Since these attributes are normally maintained as an arbitrary sequence of bytes within OVPF, it is necessary for a given feature object to have access to the byte-offset and length of each of its attributes in the *ReadWriteStream* object. This information is read and stored during metadata initialization of a VPF database; and is held in the *fileHeader* schema defined in VPFFeatureDef (for attributes defined in feature table headers) and in the *symbolHeader* schema defined in VPFFeatureDef (for text attributes defined in SYMBOL. RA).

Integer and Character Value Description Tables

One set of tables all VPF products have in common includes the *integer value description table* (INT.VDT) and the *character value description table* (CHAR.VDT). Each coverage has these VDT files to store the descriptive text associated with each of the valid ranges of feature attribute values. In OVPF, the set of all VDT data is held in a class variable of VPFFeature class called *ValueDescription*. This is organized as a nested hierarchy of tables, so that the valid range of in

eger and textual values for a given attribute is easily found. The *ValueDescriptions* structure is populated during database initialization.

Holding all of these VDTs in a single hierarchical collection structure has proven useful in identifying inconsistencies among values and descriptions across coverages, libraries, and databases. Some of these differences reflect errors in the data, but most are due to design differences in the use of a given feature attribute among different VPF products.

Feature-Related Notes

If a coverage includes a NOTES.RAT file, this is read into the VPFCoverage subclass' *featureNotes* instance variable during database initialization. A NOTES.RAT file normally has many-to-many links with feature objects; that is, any one feature may be linked to many records in the NOTES.RAT file, and any one note in the NOTES.RAT file could be linked to many different feature objects. The purpose of the *notes* instance variable defined in VPFFeature is to hold onto a list of note IDs from the NOTES.RAT file for lookup into the VPFCoverage subclass' *featureNotes* collection. The *noteHeader* schema defined in VPFCoverage is used when reading these notes from disk. So far, DNC appears to be the only VPF product using NOTES.RAT.

Graphical Primitives

Each feature object is associated with a set of latitude-longitude coordinates, referred to as graphical primitives. Point and node features are associated with entity- and connected-node primitives. Line features are associated with edge primitives. Area features are associated with a face primitive consisting of a "ring" of edge primitives, and text features are associated with a "path" of coordinates. Because any one-line feature object may consist of multiple edges, and any single node, edge, or face primitive could be used by more than one feature object, great care must be taken to maintain the correct linkage between the features and primitives. Topological relationships (adjacency and contiguity) among the primitives must also be maintained across all features within a given coverage and file, according to the VPF specification.

The design of spatial topology within OVPF is documented in [Chu+95b] and it is not an intention in this chapter to elaborate on the intricacies of such. Therefore, for purposes of this chapter, graphical primitives will be treated as if all were instances of the same class and will be called VPFDrawOrder. This approach is sufficient for explaining the manner of encoding each primitive's location coordinates. Also, VPFDrawOrder is the superclass from which all the topological primitive classes are derived in the topology framework.

As seen in Figures 15-3b and 15-4b, VPFDrawOrder objects have a pointer to a collection of feature objects. This is to allow each graphical primitive (draw order) to know and communicate directly with the feature(s) in which it is used. Figure 15-3b also shows a class variable *GraphicalPrimitives* defined for VPFDrawOrders.

This is to hold the set of all VPFDrawOrder instances imported or created in the current work session.

VPFDrawOrder objects hold all location coordinates in a specially encoded array of bytes and work much like the ReadWriteStream used to store and process a feature's attributes. A VPFDrawOrder object has an instance variable that contains the byte array. This byte array has three parts:

- Opcode byte—an 8-bit integer used to define what kind of operation is to be performed; e.g., 199 means "set polyline"
- Length byte—an 8-bit integer containing the number of data bytes to follow
- Data bytes—sequence of bytes representing a color index, location coordinates, and so on.

Instances of VPFDrawOrder are used to represent nodes, edges, the polylines forming each character in a text feature's string, and any other graphical entity associated with a feature object. These VPFDrawOrders can be concatenated into arbitrarily long sequences of bytes for any given purpose. This is a computationally efficient means of managing the storage and processing of the tens and hundreds of thousands of coordinate points that are required to represent a VPF database.

Spatial Tree Indexing Framework

Two classes shown in Figure 15-3 and 15-4, VPFSpatialDataManager and VPFSpatialDataCell, are used to implement and manage a spatial tree indexing framework. This framework presently uses a quadtree organization in which each spatial data cell holds pointers to four subcells and each subcell holds a pointer to its parent or supercell.

This spatial tree design is independent of the VPF database to be imported. All access to the spatial tree from within OVPF is done through a spatial data manager object. Each feature passed to the spatial data manager is inserted into the appropriate spatial tree cell based on the feature's minimum bounding rectangle. The spatial data manager also responds to user requests to obtain or delete the features appearing within a particular region in space.

This design allows us to modify the implementation of the spatial tree at any time without affecting the rest of OVPF or the source data. Thus, in the future, we could easily substitute an R' tree [Sto+86], PM Random (PMR) quadtree [NSS7], spatial splay tree [Cob+95], or special optimizing techniques in place of the present quadtree approach. We could also support the simultaneous implementation of multiple spatial indexing schemes to allow choice of the most efficient spatial tree design for a given source database. This may be important as we provide support for Raster Product Format (RPF) and other non-VPF databases.

DATABASE IMPLEMENTATION

The implementation of the database necessitated the following steps:

1. Making design decisions such as:
 - Deciding which objects should be transient and which objects should be persistent
 - Deciding on physical partitions of the database
 - Deciding on roots of persistent object trees
 - Establishing a security model
2. Modifying the existing implementation of OVPF in the following ways:
 - Reorganizing or creating new classes and methods needed for database use
 - Inserting correct references for persistent objects and ensuring that no persistent object references are permanently held in the application program
 - Placing transaction boundaries in appropriate places
3. Loading the OVPF database with the VPF data, including:
 - Building the initial database file
 - Importing and migrating the metadata
 - Importing and migrating the feature data

The relation of each of these steps to ObjectStore is discussed in detail in the following section.

Persistent Object Webs in OVPF

The first step of the external database design involves the decision regarding which sets of objects to make persistent. This is very different for OO than for relational DBMSs. Any object within OVPF's internal memory can become a persistent object in an ODBMS merely by "asking" it to be. However, this not only copies the requested object into the external database, it also copies the object's transitive closure of every object to which it points. However, it is often undesirable to store some objects in such a transitive closure in the external database. Each of the ODBMS products provides a means to bypass the transitive closure under certain conditions so that only selected links are followed. The implications of this issue within OVPF are discussed below.

In this section, three main groupings of database objects have been presented: the metadata object web, the feature objects, and the *ValueDescriptions* data structures. Each of these object groups is made persistent. Another category of OVPF objects includes the user interface classes. These are the support classes that present the map on the computer screen and allow interaction with the user. Instances of these classes should not be made persistent.

Typically, a complete web of objects is made persistent by reference to some "root" or "parent" object for the group. This also provides a named entry point to the persistent object web for future access by other application programs. In the case of the metadata object web, the root is a collection object containing pointers to all initialized databases, as shown in Figure 15-4a. For example, this collection has a member called DNC01 that points to its libraries, each of which points to its respective coverages, and so on. In the non-database version of OVPF, this root collection of databases is held by the VPFDatabase class variable called *Databases*. However, with the integrated database, this metadata object web is made persistent, thereby alleviating the need for the *Databases* class variable.

For the feature objects, the logical root object is the spatial tree manager that holds a pointer to the linked list of spatial tree cells, each of which holds pointers to the features whose bounding rectangle falls within the cells boundaries. Each feature object (instance of a VPFFeature subclass) holds onto its attributes stream and its symbol (instance of a VPFSymbol subclass).

For the graphic primitives, the logical root object is the collection of all VPFDrawOrders, presently held by the VPFDrawOrders class variable *GraphicPrimitives*. This root collection object is made persistent in the object database. Therefore, as was the case for the *Databases* class variable, there is no longer a need for the *GraphicPrimitives* class variable. Likewise, the *ValueDescriptions* structure held by the class variable of VPFFeature is handled in a similar manner.

ObjectStore

ObjectStore provides a relatively small set of low-level operations for database accessing and manipulation. Thus, integrating ObjectStore with OVPF was a fairly nonintrusive operation, requiring some restructuring of the OVPF class hierarchy, but few changes in fundamental design or implementation.

Conceptual Database Organization

One of the first changes made was a reorganization of the OVPF class hierarchies to change the usage of class variables. ObjectStore does not allow objects held by these kinds of variables to be made persistent; instead, all persistent states should be held by instance variables. For example, as discussed earlier, OVPF previously used class variables of the graphical primitive classes to hold centralized collections of all primitives organized by coverage. This collection was moved and divided to be held by instance variables of each coverage object so that each coverage now holds a collection of just its own primitives. Another class variable previously held the root of a tree structure of all value description table (VDT) entries in a given database. This collection was also moved and divided so that each features metadata (called its featureDef) object now holds a collection of just the VDT entries for that feature class.

Physical Database Design

ObjectStore facilitates clustering of data through the use of *segments*. A segment is a variable-sized region of storage composed of disk pages. Segments or pages can be specified as the unit of transfer of data from disk to memory, and can range in size from the default of 4 KB to over 128 MB. Unless otherwise directed, all data are automatically stored in a predefined segment called the *default segment*. However, it is usually desirable to create multiple segments, as the appropriate designation of segments for specific groups of objects can dramatically improve database performance by defining which objects are transferred together.

OVPF uses four segments, known as the *default segment*, the *spatial segment*, the *feature segment*, and the *primitive segment*. Each segment stores a group of related objects as given below:

- **Default segment**—the database metadata
- **Spatial segment**—data relating to the spatial index (quadtree)
- **Feature segment**—all feature-level data
- **Primitive segment**—all graphic primitive data; for example, edge, face, node data

This physical partitioning of the data was performed to support an optimal clustering strategy. This helps prevent unnecessary data from being transferred upon access to the database. Very little change was required in OVPF to define and access these segments.

ObjectStore allows multiple entry points to each database to provide efficient access to various logical hierarchies of data. Each entry point is referred to as a *named root* of persistent data. Each object designated as a root object can be accessed directly through the use of its name. All objects in the root objects composition hierarchy can then be accessed by navigating through its instance variables.

Two root objects were created for the OVPF database. These are *OSDatabasesRoot*, which provides access to the different VPF databases such as *WVS+* and *DNC*, and *OSSpatialIndexesRoot*, which provides an entry point for the quadtree spatial indexing structure. In OVPF, the spatial indexing organization was modified somewhat so that each coverage has a pointer to its own quadtree in the database. Direct access to each quadtree is essential for fast responses to user queries, and the *OSSpatialIndexesRoot* provides access to the roots of all quadtrees in the current database. While the term *quadtree* is used in this chapter, OVPF is organized to support any combination of different types of spatial indices among VPF products, with each feature coverage specifying its own spatial index. Figure 15-5 shows the four OVPF database segments and their connections to each other. Database roots are noted through the use of the symbol.

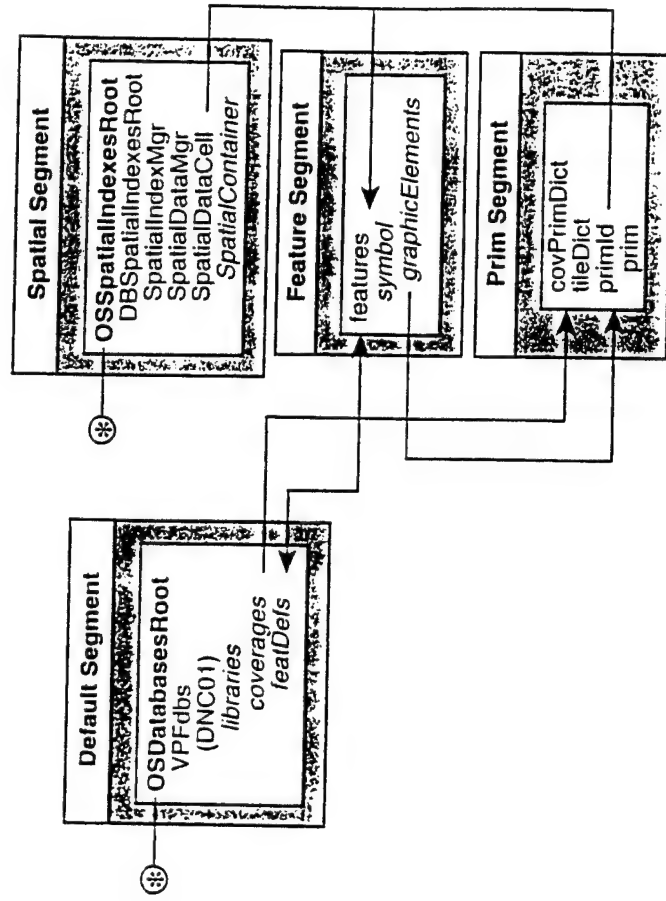


Figure 15-5 OVPF ObjectStore database segments.

Proxies and OSReferences

The class *OSReference* is provided by *ObjectStore* to handle inter-object references from transient to persistent objects. An instance of the *OSReference* class replaces a reference from a transient to a persistent object upon transaction commit. *OSReferences* provide object identifiers that are valid across transactions. *OSReferences* forward messages within a transaction to the persistent objects they represent and as such can be used in a manner similar to that of the persistent objects themselves. *OSReferences* were used in OVPF for instances of the metadata classes *VPFDatabase*, *VPFLibrary*, *VPFCoverage*, and *VPFFeatureDef*.

Proxies, implemented by the lightweight *OSProxy* class, are used when a handle to a persistent object is needed, but when references to that object do not necessarily require all of the objects data. A string can be assigned to a proxy object such that references to the persistent object represented by the proxy that occur outside of transactions will return the proxy's string. Any references made to the proxy while inside a transaction will return the persistent object. Figure 15-6 illustrates the use of proxies with OVPF.

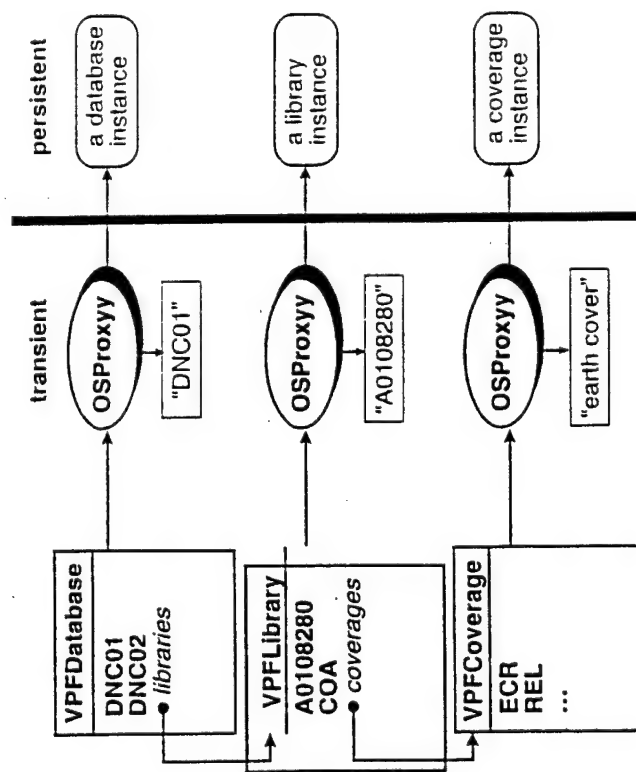


Figure 15-6 Use of proxies with OVPF.

Loading the Database

Once the database file has been created through use of the appropriate ObjectStore method, it may be loaded with the VPF data. This is done in two steps. First, the metadata is imported from the VPF tables (metadata includes VPF tables such as the Database Header Table (DHT), Library Attribute Table (LAT), and Library Header Table (LHT), each feature class's schema, and the VPFs). After the metadata have been loaded and migrated to the database, feature data are imported from the VPF database, one coverage at a time, and placed in the memory-based quadtree spatial indexing structure. The features and the quadtree are then migrated to the database in their respective segments upon commit of the feature loading transaction. After the bulk loading of the databases in this manner, the relational tables are not used for subsequent data access—all manipulation of the data involves only the object database.

EXPERIENCES AND SUMMARY

The integration of ObjectStore with OVPF has proven to be promising. One of the major accomplishments in this effort is that an ODBMS was integrated as a layer to the OVPF model. In other words, major changes to the OVPF model were not re-

quired for integration with ObjectStore. It is possible that OVPF design and implementation are stable and robust to support add-on modules. However, this needs to be explored more.

With a database management system, our prototype has shown improvements both in performance and in the design. Modifications had to be made to the OVPF model to integrate it with ObjectStore. However, these were minor changes. Eliminating all class variables in the model—since ObjectStore does not support persistent class variables—and using more instance variables to hold onto the information held by class variables took some effort. This is an area where ObjectStore could have made the migration process a little simpler by providing support for persistent class variables. On the other hand, changes in the design allowed each coverage to have a unique spatial indexing scheme. Before the integration, only one spatial indexing scheme was used for all features. This change promotes the possibility of using the spatial indexing scheme that is most optimal for a specific coverage, which should provide faster access and retrieval.

ObjectStore required use of cut-points to prevent pulling the graphic user interface into the internal database. From an application design standpoint, this improved the modularity of our code and helped us identify places where actual feature or primitive objects were sending messages directly to the map window objects. In our design, the map GUI object is allowed to talk to the features and primitives, but not vice versa.

With a spatial segment, the spatial indexing scheme as implemented in OVPF was migrated to the external database without any modification. This was possible since the spatial indexing scheme or quadtree is implemented as an independent class that contains feature-object pointers. This allows fast, direct lookup of geographic features in response to queries.

With the prototype, overall performance has improved in the areas of data import, display, and querying. There are two contributing factors to these improvements. First, ObjectStore uses the CacheManager to cache information retrieved from the internal database. Secondly, direct navigation to the object web is supported via clustering; therefore, resolution of spatial queries is fast and direct.

Furthermore, a sequential selection dependency for import no longer exists. Without ObjectStore, a database must be selected, then a library, then a coverage, followed by a selection of import type (e.g., point, line, area, or all feature types). If a user wants to display information from two different databases, one database has to be imported and displayed first before another database can be brought in. However, this is no longer true after the integration. All of the database can be selected down to the coverage level at once for an aggregate display on screen.

Aside from these improvements in the OVPF model, the pure advantage of having a database management system is a gain in itself. The concept of persistent objects allows accessing data without having to load the data into memory as in the model before the integration. Multi-user access and configuration control are provided to allow more use of the data without having to maintain a copy of data on every user's computer and subsequently having to be concerned with data consistency.

For future work, we have to address fast retrieval for queries based on values of feature attributes. This is a common type of query, but unfortunately we have not indexed any of the attributes of features. The first level of support for this kind of query from the ODBMS is to use an effective clustering strategy, so perhaps features will be organized on disk on some order based on attribute values. This is not very flexible, though. A better approach will be to create and manage B-trees or hash tables for fast lookup of objects based on a given attribute. To facilitate this, it will probably be necessary to make real instance variables for each attribute to be indexed.

REFERENCES

- [Arc+95c] D. K. Arctur, E. Anwar, J. F. Alexander, S. Chakravarthy, M. J. Chung, M. A. Cobb, K. B. Shaw: *Comparison and Benchmarks for Import of Vector Product Format (VPF) Geographic Data from Object-Oriented and Relational Database Files*. Proc. Fourth Symp. on Spatial Databases (pp. 368-384). New York: Springer-Verlag, 1995.
- [Arc+95a] D. K. Arctur, J. F. Alexander, M. A. Cobb, M. J. Chung, K. B. Shaw: *OVPF Chapter: Evaluation of Illustra Hybrid Object-Relational DBMS*. Naval Research Laboratory, Stennis Space Center, MS, 1995.
- [Arc+95b] D. K. Arctur, J. F. Alexander, M. A. Cobb, M. J. Chung, K. B. Shaw: *OVPF Report: Issues and Approaches for Spatial Topology in GIS*. NRL/MR/7441-96-7719, Naval Research Laboratory, Stennis Space Center, MS, 1995.
- [Chu+95a] M. J. Chung, M. A. Cobb, K. B. Shaw, D. K. Arctur, J. F. Alexander: *OVPF Chapter: Network Investigation Results*. NRL/MR/7441-95-7713, Naval Research Laboratory, Stennis Space Center, MS, 1995.
- [Chu+95b] M. J. Chung, M. A. Cobb, K. B. Shaw, D. K. Arctur: *An Object-Oriented Approach for Handling Topology in VPF Products*. Proc. GIS/LIS '95, Nashville, TN, Vol. 1, pp. 163-174, 1995.
- [Cob+95] M. A. Cobb, M. J. Chung, K. B. Shaw, D. K. Arctur: *A Self-Adjusting Indexing Structure for Spatial Data*. Proc. GIS/LIS '95, Nashville, TN, Vol. 1, pp. 182-192, 1995.
- [DMA92] Defense Mapping Agency: *Military Standard: Vector Product Format*. Draft Document No. MIL-STD-2407, Defense Mapping Agency, Fairfax, VA, 1992.
- [NS87] R. C. Nelson, H. Sanet: *A Population Analysis for Hierarchical Data Structures*, in Proceedings of the ACM SIGMOD Conference on Management of Data, San Francisco, CA, pp. 270-277, 1987.
- [Sha+95] K. B. Shaw, M. J. Chung, M. A. Cobb, D. K. Arctur, J. F. Alexander, E. Anwar: *Object-Oriented Database Exploitation Within the GIS Data*

Warehouse: Initial Chapter. NRL/FR/7441-95-9639, Naval Research Laboratory, Stennis Space Center, MS, April 1995.

[Sto+86] M. T. Stonebraker, T. Sellis, E. Hanson: *An Analysis of Rule Indexing Implementations in Data Base Systems*, in Proceedings of the First International Conference on Expert Database Systems, Charleston, SC, pp. 353-364, April 1986.

2002 Annual Meeting of The North American Fuzzy Information Processing Society Proceedings

NAFIPS-FLINT 2002 International Conference

Tulane University, New Orleans LA

June 27-29, 2002

20 Years of NAFIPS

LOGAN, UTAH

1982

TO

NEW ORLEANS, LOUISIANA

2002



IEEE

02TH8622

**2002 Annual Meeting of the North American
Fuzzy Information Processing Society
Proceedings**

NAFIPS-FLINT 2002

**June 27-29, 2002
Tulane University
New Orleans, Louisiana**

Edited by

**Jim Keller
Olfa Nasraoui**

Sponsored by

IEEE Systems, Man and Cybernetics Society

02TH8622

Fuzzy Spatial Querying with Inexact Inference

Huiqing Yang, Maria Cobb

Dia Ali, Shahram Rahimi

Department of Computer Science & Statistics

University of Southern Mississippi

Hattiesburg, Mississippi, USA 39406-5601

hyan2@orca.st.usm.edu

Maria.Cobb@usm.edu

dia.ali@usm.edu

shahram.rahimi@usm.edu

Frederick E. Petry

Electrical Engineering & Computer Science

Tulane University, New Orleans, LA 70118

petry@eecs.tulane.edu

Kevin B. Shaw

Naval Research Laboratory Mapping,

Charting & Geodesy

Stennis Space Center, MS, USA 39529

shaw@nrlssc.navy.mil

Abstract

The issue of spatial querying accuracy has been viewed as critical to the successful implementation and long-term viability of the GIS technology. In order to improve the spatial querying accuracy and quality, the problems associated with the areas of fuzziness and uncertainty are of great concern in the spatial database community. There has been a strong demand to provide approaches that deal with inaccuracy and uncertainty in GIS. In this paper, we are dedicated to develop an approach that can perform fuzzy spatial querying under uncertainty. An inexact inferring strategy is investigated. The study shows that the fuzzy set and the certainty factor can work together to deal with spatial querying. Querying examples implemented by FuzzyClips are also provided.

Keywords: uncertainty, inexact inferring, fuzzy inference, spatial query, GIS, FuzzyClips

1. Introduction

Since the spatial querying deals with some concepts expressed by verbal language, the fuzziness is frequently involved. Hence, the ability to query a spatial data under the fuzziness is one of the most important characteristics of any spatial databases. Some researchers have shown that the directional as well as topological relationships are fuzzy concepts [1-2]. To support queries of this nature, our earlier works [3-6] provided a basis for fuzzy querying capabilities based on a binary model. The Clips-based implementation [6] shows the fuzzy querying can distinguish various cases in the same relation classes. For instance, consider the example relationship *Object A overlaps Object B*. The

fuzzy querying can answer: does all of *Object A* overlap some of *Object B*, or does little of *Object A* overlap most of *Object B*?

However, in these kinds of fuzzy queries, the representation of the fuzzy variables is based on classical set theory. Although classical sets are suitable for various applications and have proven to be an important tool for mathematics and computer science, they do not reflect the nature of human concepts and thoughts, which tend to be abstract and imprecise. The flaw comes from the sharp transition between inclusion and exclusion in a set. In this paper we show a way to use the fuzzy set for dealing with the vague meaning of linguistic terms, in which the smooth transition is characterized by membership function.

The queries expressed by verbal language often involve a mixture of uncertainties in the outcomes that are governed by the meaning of linguistic terms. Therefore, there is an availability-related need for skilled inexact inferring approach to handle the uncertain feature [7]. Uncertainty occurs when one is not absolutely certain about a piece of information. Although uncertainty is an inevitable problem in spatial queries, there are clear gaps in our understanding of how to incorporate uncertain reasoning into the spatial querying process. This requires performing an inexact inferring. Recently, models of uncertainty have been proposed for spatial information that incorporate ideas from natural language processing, the value of information concept, non-monotonic logic and fuzzy set, evidential and probability theory. Each model is appropriate for a different type of inexactness in spatial data. By incorporating the fuzzy set and confirmation theory, we investigate an inexact inferring approach

for fuzzy spatial querying. The aim is to improve spatial querying accuracy and quality.

The paper is organized as follows. Section 2 briefly overviews our previous works, and shows some basic techniques and strategies to deal with fuzzy multiple relations in spatial querying. Section 3 describes an approach that can perform fuzzy querying under the uncertainties. In section 4, FuzzyCLIPS implementation shows some improved querying results.

2. An Overview of Previous Works

Assume that the spatial objects can be approximated by their minimum bounding rectangles (MBR). Figure 1 shows two objects in two dimensions. Based on the spatial binary model [3-6], some spatial querying techniques and strategies can be briefly overviewed as follows.

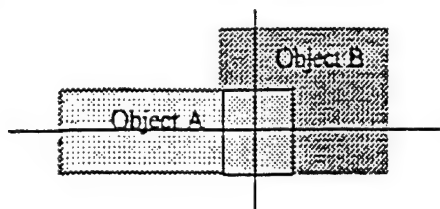


Figure 1. Two objects in 2-D

2.1 Basic Spatial Querying

Topological and directional relationships are critical components in the retrieval of information from spatial databases, including image, map and pictorial databases. Many contributions have been made. The authors in [10] define new families of fuzzy directional relations in terms of the computation of force histograms, which is based on the raster data. In this paper, we will take into account these two major spatial relations based on the vector data.

The topological relationships express the concepts of inclusion and neighborhood. A large body of related work has focused on the intersection mode that describes relations using intersections of object's interiors and boundaries. By means of geometrical similarity, we defined the topological relationships as a set:

$$T = \{\text{disjoint, tangent, surround, overlaps, ...}\}.$$

The paper [3] provides greater details on this.

The directional relationships are commonly concerned in everyday life. Most common directions are cardinal direction and their refinement. We defined the directional relations as a following set:

$$D = \{\text{North, East, South, West, Northeast, Southeast, Southwest, Northwest}\}.$$

Such relationships provided a significant resource for the basic binary spatial queries. The examples of such queries might look like these:

Object A overlaps Object B.

Object A is west of Object B.

2.2 Fuzzy Spatial Querying

Although the above querying method can provide topological and directional information, these kinds of information do not associated with any degrees. This means it can only perform a low level query. A typical example is shown in Figure 2.



Figure 2.

For both cases that belong to the same class (or relation group), the basic spatial querying will provide the same topological and directional relationships, i.e. Object A overlaps Object B and Object A is west of Object B.

How to provide high accurate information, such as most of Object A overlaps some of Object B, or little of Object A overlaps some of Object B and so on, encourages us to make the further investigation. Some strategies and techniques can be briefly described as follows (see the details in [6]).

- Partition each object into sub-groups in eight directions based on the reference area (the common part of two objects) shown in Figure 3;
- Map each sub-group to a node, and assign two weights (area and node weights) to each node;
- Calculate two weights to determine the special degree.

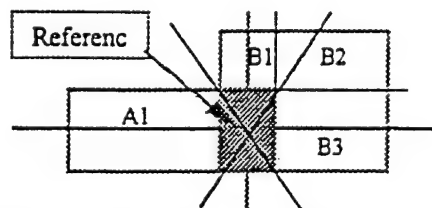


Figure 3. Partitioning two objects in 2D

Where area weight can be calculated by

$$AW = (\text{area of sub-group}) / (\text{area of the entire object})$$

and node weight can be obtained by

$$NW = AW \cdot (\text{axis length}) / (\text{longest axis length}).$$

In order to support fuzzy querying, the resulting quantitative figures (AW, NW) are mapped to a range that corresponds to a term known as linguistic qualifiers. There is a huge body of knowledge and techniques that deal with fuzzy spatial relations in linguistic expression. In this paper, we define the topological qualifier TQ and directional qualifier DQ as:

TQ={all, most, some, little, none}

DQ={directly, mostly, somewhat, slightly, not}.

As mentioned in [9], relative qualifiers can be represented as fuzzy subsets of the unit interval and use linguistic word. Based on the classical set, the membership function of qualifiers can be defined as a binary set, that is, complete membership has a value of 1, and no membership has a value of 0. The following tables give the quantifying description.

Table 1. Topological Qualifiers

Topological Qualifiers (TQ)	Area Weight (AW)
all	0.96 to 1.00
most	0.60 to 0.95
some	0.30 to 0.59
little	0.06 to 0.29
none	0.00 to 0.05

Table 2. Directional Qualifiers

Directional Qualifiers (DQ)	Node Weight (NW)
directly	0.96 to 1.00
mostly	0.60 to 0.95
somewhat	0.30 to 0.59
slightly	0.06 to 0.29
not	0.00 to 0.05

As shown in Figure 1, the based-Clips implementation can provide the following information:

Most of Object A overlaps Object B
Object A overlaps some of Object B

Most of Object A overlaps some of Object B

Most of Object A is west of Object B
Object A is mostly west of Object B

Most of Object A is mostly west of Object B

3. Fuzzy Querying Under Uncertainty

Because the spatial relationships depend on human interpretation, spatial querying should be related by fuzzy concepts. To support queries of the nature,

previous works provided fuzzy queries without uncertainty that can handle the fuzziness by defining fuzzy qualifiers. However, in these kinds of fuzzy queries, the particular grades of membership have been defined as classical sets. The problem is there exist a gap between two neighboring members such as 'all' and 'most'. Because a jump occurs, no qualifier is defined in some intervals, for example the interval (0.95, 0.96). To improve the fuzzy querying, the fuzzy set theory is concerned in our continuous research.

3.1. Fuzziness Consideration

Fuzziness occurs when the boundary of a piece of information is not clear-cut. Hence, fuzzy querying expands query capabilities by allowing for ambiguity and partial membership. The definition of the grades of membership is subjective and depends on the human interpretation. A way to eliminate subjectivity is another interested research field. Here simple membership functions will be considered.

A fuzzy set is a set without a crisp boundary. The smooth transition is characterized by membership functions that give fuzzy sets flexibility in linguistic expressions. More formally a fuzzy set in a universe is characterized by a membership function $\mu: U \rightarrow [0,1]$. Figure 4 illustrates the primary term of fuzzy variable area weight. Each term represents a specific fuzzy set.

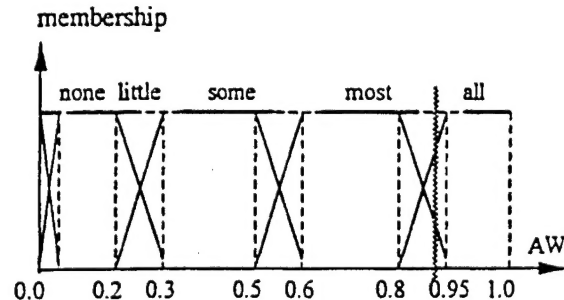


Figure 4. Membership function for TQ

The fuzzy set functions for topological qualifiers can be described as:

$$\mu_{all}(AW) = \begin{cases} 1.0 & \text{if } 0.95 \leq AW \leq 1.0 \\ 20(AW - 0.80)/3 & \text{if } 0.8 \leq AW \leq 0.95 \end{cases}$$

$$\mu_{most}(AW) = \begin{cases} 20(0.95 - AW)/3 & \text{if } 0.8 \leq AW \leq 0.95 \\ 1.0 & \text{if } 0.6 \leq AW \leq 0.80 \\ 10(AW - 0.5) & \text{if } 0.5 \leq AW \leq 0.6 \end{cases}$$

$$\mu_{some}(AW) = \begin{cases} 10(0.6 - AW) & \text{if } 0.5 \leq AW \leq 0.6 \\ 1.0 & \text{if } 0.3 \leq AW \leq 0.5 \\ 10(AW - 0.2) & \text{if } 0.2 \leq AW \leq 0.3 \end{cases}$$

$$\mu_{\text{little}}(AW) = \begin{cases} 10(0.3 - AW) & \text{if } 0.2 \leq AW \leq 0.3 \\ 1.0 & \text{if } 0.02 \leq AW \leq 0.2 \\ 100(AW - 0.01) & \text{if } 0.01 \leq AW \leq 0.02 \end{cases}$$

$$\mu_{\text{none}}(AW) = \begin{cases} 100(0.02 - AW) & \text{if } 0.01 \leq AW \leq 0.02 \\ 1.0 & \text{if } 0.0 \leq AW \leq 0.01 \end{cases}$$

In the same way, the fuzzy set functions for directional querying can be described as:

$$\mu_{\text{directly}}(NW) = \begin{cases} 1.0 & \text{if } 0.95 \leq NW \leq 1.0 \\ 20(NW - 0.80)/3 & \text{if } 0.8 \leq NW \leq 0.95 \end{cases}$$

$$\mu_{\text{mostly}}(NW) = \begin{cases} 20(0.95 - NW)/3 & \text{if } 0.8 \leq NW \leq 0.95 \\ 1.0 & \text{if } 0.6 \leq NW \leq 0.8 \\ 10(NW - 0.5) & \text{if } 0.5 \leq NW \leq 0.6 \end{cases}$$

$$\mu_{\text{somewhat}}(NW) = \begin{cases} 10(0.6 - NW) & \text{if } 0.5 \leq NW \leq 0.6 \\ 1.0 & \text{if } 0.3 \leq NW \leq 0.5 \\ 10(NW - 0.2) & \text{if } 0.2 \leq NW \leq 0.3 \end{cases}$$

$$\mu_{\text{lightly}}(NW) = \begin{cases} 10(0.3 - NW) & \text{if } 0.2 \leq NW \leq 0.3 \\ 1.0 & \text{if } 0.02 \leq NW \leq 0.2 \\ 100(NW - 0.01) & \text{if } 0.01 \leq NW \leq 0.02 \end{cases}$$

$$\mu_{\text{not}}(NW) = \begin{cases} 100(0.02 - NW) & \text{if } 0.01 \leq NW \leq 0.02 \\ 1.0 & \text{if } 0.0 \leq NW \leq 0.01 \end{cases}$$

Unlike classical set theory that can describe membership to a set clearly, in fuzzy set theory membership of a term to a set is partial, i.e., a term belongs to a set with a certain grade of membership. Although it solves the gap problem in classical set expression, a new problem is coming. Because a common feature of the fuzzy sets is overlapping, the qualifiers may be associated with two different terms at the intersect intervals. For instance, the topological qualifier TQ may take 'all' and 'most' at the same time. This reveals uncertainty – the lack of adequate and correct information to make a decision.

3.2. Uncertainty Consideration

Uncertainty is an inevitable problem in GIS. In this paper, we devote ourselves to explore an approach that can perform the fuzzy querying under uncertainties. The study exemplifies whether the fuzzy set and certainty factor can incorporate in spatial querying.

Uncertainty occurs when one is not absolutely certain about a piece of information. Given $AW=0.90$, the fuzzy querying may give the following querying phrase:

All of Object A overlaps Object B
Most of Object A overlaps Object B.

How do we make the decision according to the information? Which querying information is reliable?

This reveals important deficiencies in areas such as the reliability of queries and the ability to detect inconsistencies in the knowledge. Because we cannot be completely certain that some qualifiers are true or others are false, we construct a certainty factor (CF) to evaluate the degree of certainty. The degree of certainty is usually represented by a crisp numerical value one, a scale from 0 to 1. A certainty factor of 1 indicates that it is very certain that a fact is true, and a certainty factor of 0 indicates that it is very uncertain that a fact is true. Some key ideas relevant to the determination the CF are discussed as following.

Case 1. Considered a single qualifier for each querying

This is a case in which only one qualifier associated with a single object is involved in each querying result such as:

All of Object A overlaps Object B

Object A is directly west of Object B.

Where the fuzzy topological qualifier TQ_A = 'all' which is associated with the object A; the fuzzy directional qualifier DQ_A = 'directly' which is associated with the object A.

- If the qualifier only takes one term at given interval, the grade of membership $\mu(\cdot)$ can be used as a CF that represents the degree of belief. The results will look like:

All of Object A overlaps Object B

with $CF = \mu_{\text{all}}(AW_{ai} = 0.99) = 1.0$

Object A is directly west of Object B

with $CF = \mu_{\text{directly}}(NW_{ai} = 0.99) = 1.0$

Where AW_{ai} is the area weight of a sub-group associated with object A; NW_{ai} is the node weight of a subgroup associated with object A; and $i, j \in [1, 8]$, i represents an integer set.

- If the given weight is in the overlapping area, two qualifiers will be related. For example, the fuzzy topological qualifier of the object A takes both 'all' and 'most'. The querying results will be:

All of Object A overlaps Object B

Most of Object A overlaps Object B

It is acceptable if we take the qualifier that has a larger grade of membership. The certainty factor can be determined by the maximum value, that is,

$$CF = \max\{\mu_{\text{all}}(AW_{ai} = 0.90), \mu_{\text{most}}(AW_{ai} = 0.90)\} \\ = \mu_{\text{all}}(AW_{ai} = 0.90).$$

The final querying results should be
All of Object A overlaps Object B
 with $CF = \mu_{all}(AW_{ai} = 0.90)$.

As a result, the CF in case 1 can be obtained by

$CF = \max\{\mu_{TQ_k}(AW_{ai} = \text{const}), k \in I[1,5], i \in I[1,8]\}$
 $CF = \max\{\mu_{DQ_k}(NW_{aj} = \text{const}), k \in I[1,5], j \in I[1,8]\}$
 Where
 TQ_k is a topological qualifier such as all;
 DQ_k is a directional qualifier such as directly;
 AW_{ai} is an area weight associated object i-node
 NW_{aj} is a node weight associated object j-node
 * is used to represent object A or B

Case 2. Considered multiple qualifiers

In the querying results, many pieces of fuzzy terms are conjoined (i.e. they are joined by AND), or disjoined (i.e. joined by OR). The examples of these types of queries are as follows:

Most of Object A overlaps some of Object B
Some of Object A is slightly south of Object B.

Hence, to perform these kinds of queries, we have to handle multiple fuzzy qualifiers. It is easy to understand that the relationship between different object qualifiers is conjunction, and the relationship between the same object qualifiers is disjoined. According to the fuzzy set theory, the conjunction and disjunction of fuzzy term can be respectively defined as the minimum and maximum of the involved facts. Therefore, the certainty factor contained multiple qualifiers can be determined by the following formulas:

Consider topological relationships
 $CF = \min\{\max\{\mu_{TQ_{ka}}(AW_{ai} = \alpha)\}, \max\{\mu_{TQ_{kb}}(AW_{bj} = \alpha)\},$
 where $ka, kb \in I[1,5]$ & $i, j \in I[1,8]\}$
 Note: a topological qualifier $TQ_i = \text{all}$ if $ka = 1$.

 Consider topological/directional relationships
 $CF = \min\{\max\{\mu_{TQ_{ka}}(AW_{ai} = \alpha)\}, \max\{\mu_{DQ_{ka}}(NW_{aj} = \beta)\},$
 where $ka \in I[1,5]$ & $i, j \in I[1,8]\}$
 where a_i, b_j represent object node associated with object A and B, respectively,
 α and β are constant.

As seen above, an approach in which the fuzzy set and uncertainty can incorporate to perform the fuzzy queries is developed.

4. FuzzyCLIPS Implementation

FuzzyCLIPS is an enhanced version of CLIPS developed at the National Research Council of Canada to allow the implementation of fuzzy expert systems. The modifications made to CLIPS contain the capability of handling fuzzy concepts and reasoning. It allows any mix of fuzzy and normal terms, numeric-comparison logic controls, and uncertainties in the rule and facts. By using FuzzyClips, it is easy for us to deal with fuzziness in approximate reasoning, to manipulate uncertainty in the rules and facts.

In the process of our implementation, all fuzzy variables are predefined with the *deftemplate* statement. This is an extension of the standard *deftemplate* construct in CLIPS. For example, fuzzy variables (qualifiers) can be declared in *deftemplate* constructs as following:

```

(deftemplate TFVariable
  0 1; define the fuzzy variable area-weight
  ((all (0.8 0.0)(0.95 1.0)(1.0 1.0))
   (most (0.5 0.0)(0.6 1.0)(0.8 1.0) (0.95 0.0))
   (some (0.2 0.0)(0.3 1.0)(0.5 1.0) (0.6 0.0))
   (little(0.01 0.0)(0.02 1.0)(0.2 1.0) (0.3 0.0))
   (none (0.0 1.0)(0.01 1.0)(0.02 0.0)) ) )

(deftemplate DFVariable
  0 1; define the fuzzy variable node-weight
  ((directly (0.8 0.0)(0.95 1.0)(1.00 1.0))
   (mostly (0.5 0.0)(0.60 1.0)(0.80 1.0)(0.95 0.0))
   (somewhat(0.2 0.0)(0.30 1.0)(0.50 1.0)(0.60 0.0))
   (slightly (0.01 0.0)(0.02 1.0)(0.20 1.0)(0.30 0.0))
   (not (0.0 1.0)(0.01 1.0)(0.02 0.0)) ) )
  
```

A number of commands supplied in FuzzyCLIPS are very helpful for user to access fuzzy components that they need. In our application, when the weights (fuzzy variables) are calculated, the only interested information is the value of the fuzzy set at the specified weight value. The command *get-fs-value* provides us a tool to access the value. The syntax of the command is:

```

(get-fs-value ?<fact-variable> <number>) or
(get-fs-value <integer> <number>) or
(get-fs-value <fuzzy-value> <number>),
  
```

where <number> is a value that must lie between the lower and upper bound of the universe of discourse for the fuzzy set. A simple example just look like:

```

(assert (TFVariable most))
(defrule Get-CF
  ?f <- (TFVariable ?cf)
  => (printout t "CF for " ?cf " is " (get-fs-value ?f AW)
      crlf) (retract ?f)
)
  
```

5. Querying Examples

Given two objects A(1, 1) (7, 2) and B(2, 1)(9, 4). The previous works based on CLIPS will provide the following query information.

```

=====
Query results of binary
spatial relationships
=====
2D physical relations: A |os| B.
Topological relations: A {overlaps} B.
Directional relations: A {South} B
                      A {South-West} B
                      A {west } B

Little of Object A is West of Object B
Object A is slightly West of Object B
=====
=> Little of A is slightly West of B.

```

Based FuzzyCLIPS, the querying results would be look like:

```

=====
Fuzzy Query results with certainty factor
=====
Topological information:
83% of A overlaps 23.8% of B
Most of A overlaps some of B with CF=0.778

Directional information:
Little of A is West of B with CF = 1.0
A is slightly West of B with CF = 1.0
=====
⇒ Little of A is slightly West of B
with CF= 1.0

```

More details for analysis are provided as following. Table 3 shows part of quantitative information stored in nodes associated with object.

Table 3. Quantitative information

Object Name	Node	AW	NW
Object A	center	0.8333	
	west	0.1667	0.1667
Object B	center	0.2380	
	north	0.4762	0.2655
	:	:	:

From these data, we know

$AW_{a0} = 0.8333$, TQ → { all, most};

$AW_{a7} = 0.1667$, TQ → { little};

$AW_{b0} = 0.4762$, TQ → { some};

$NW_{a7} = 0.1667$, DQ → { slightly}.

$$\left. \begin{array}{l} \mu_{all}(AW_{a0}=0.8333)=0.222 \\ \mu_{most}(AW_{a0}=0.8333)=0.778 \\ \mu_{some}(AW_{b0}=0.4762)=1.0 \end{array} \right\} \begin{array}{l} \max=0.778 \\ \\ \end{array} \left. \right\} \min=0.778$$

$$\left. \begin{array}{l} \mu_{little}(AW_{a0}=0.1667)=1.0 \\ \mu_{slightly}(NW_{b0}=0.1667)=1.0 \end{array} \right\} \min=1.0$$

6. Conclusion

In a real world, fuzziness and uncertainty can occur simultaneously. To improve spatial querying accuracy, our research investigates an inexact inferencing approach that can perform fuzzy querying under uncertainty. The reliability of querying information is judged by a certainty factor (CF). The improved fuzzy querying is very flexible, and it can return spatial information in a wider variety of forms.

7. Acknowledgement

This research was supported in part by the Naval Research Laboratory's Base Program, Program Element Number 0602435N.

References

- [1]. E. Takahashi, N. Shima, and F. Kishino, "An Image Retrieval Method Using Inquires on Spatial Relationships", *Journal of Information Processing*, 15(3), 1992, pp441-449.
- [2]. S. Winter, "Topological Relations between Discrete Regions", *SSD '95*, Portland, ME, USA, August 1995, pp310-327.
- [3]. M. A. Cobb, "Modeling Spatial Relationships within a Fuzzy Framework", *Journal of the American Society for Information Science*, 49(3): 253-266, 1998.
- [4]. M. A. Cobb, "An approach for the Definition, Representation and Querying of Binary Topological and Directional Relationships Between two-dimensional Objects", PhD thesis, Tulane University, 1995.
- [5]. M. A. Cobb, F. E. Petry, "Fuzzy Querying of Binary Relationships in Spatial Databases", 1995 IEEE, pp3624-3629.
- [6]. H. Yang, M. Cobb, K. Shaw, "A Clips-Based Implementation for Querying Binary Spatial Relationships", *Proceedings of Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, Vancouver, British Columbia, Canada, July 25-28, 2001.
- [7]. M. Goodchild, S. Gopal, "The accuracy of Spatial Databases", Basingstoke, UK: Taylor and Francis, 1990.
- [8]. S. D. Bruin, "Querying Probabilistic Land Cover Data Using Fuzzy Set Theory", *GIS*, Vol 14, No. 4.
- [9]. L.A. Zadeh, "Fuzzy Sets", *Information and Control*, 8:338-353, 1965.
- [10]. P. Matsakis, J.M. Keller, L. Wendling, J. Marjamaa, and S. Sjahputera, "Linguistic Description of Relative Positions of Objects in Images", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 31, No. 4, 2001, pp573-588.